UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

Software de desarrollo de destrezas específicas para personas con Síndrome de Down

Federico Alonso Pallarés
Tutor: Pilar Rodríguez Marín

JUNIO 2016

Software de desarrollo de destrezas específicas para personas con Síndrome de Down

AUTOR: Federico Alonso Pallarés
TUTOR: Pilar Rodríguez Marín

Dpto. Ingeniería Informática
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Junio de 2016

Resumen

Este Trabajo Fin de Grado ha consistido en la realización de la aplicación *Juegamáticas*. Se trata de una aplicación en la nube, accesible para cualquier usuario desde cualquier dispositivo con acceso a internet.

El objetivo de *Juegamáticas* es ayudar a las personas con Síndrome de Down a apoyar y reforzar su aprendizaje y conocimientos matemáticos empleando un sistema de *gamificación* guiado. Para ello, la aplicación presenta diferentes tipos de ejercicios que van escalando en dificultad según el usuario responda correcta o incorrectamente.

Al ser un proyecto multidisciplinar, desde el primer momento se ha trabajado en el desarrollo de la aplicación en colaboración con miembros docentes del colegio de educación especial *María Corredentora* de Madrid. Junto a ellos, hemos aplicado diferentes metodologías tanto para el desarrollo de las actividades (tipos de ejercicio, niveles, orden de los mismos, etc.), como para el de la propia aplicación (usabilidad, estilos, elementos de ayuda y guiado, etc.). También hemos colaborado directamente con los educadores del colegio para el diseño de la interacción entre los usuarios y la aplicación (refuerzos positivos, penalizaciones, etc.).

El proyecto se ha desarrollado utilizando la plataforma *Google App Engine* para desarrollo de aplicaciones web. Los datos se almacenan en un espacio proporcionado por *Google Data Store*, y se gestionan mediante *Objectify* (también de *Google*).

Con todo ello, y una vez finalizada esta fase de desarrollo, hemos concluido que los resultados son satisfactorios, y permitirán seguir avanzando en esta línea para conseguir que *Juegamáticas* esté a disposición de cualquier persona que considere utilizarlo como apoyo al aprendizaje matemático.

Palabras clave

Aprendizaje, refuerzo, gamificación, educación, tutor, estudiante, Síndrome de Down, sistema de evaluación, Google App Engine, datastore, Objectify, ejercicios de asociación, ejercicios de operación.

Abstract

The present end of degree work is based on the implementation of the application *Juegamáticas*. It consists on a cloud application, accessible for any user using any device with internet access.

The aim of "Juegamáticas" is to help people with *Down Syndrome* by supporting and reinforcing their learning process and mathematical knowledge using a guided gamification system. To do this, the application presents different types of exercises, which escalate in difficulty depending on the correct or incorrect answer of the user.

Being, as it is, a multidisciplinary project, from the first moment we have worked in the development of the application in collaboration with faculty members from the special education school centre *María Corredentora* (Madrid). Along with them, we have applied different methodologies for the development of the different activities (types of exercises, levels, order of these, etc.), the development of the application (usability, styles, help and guide elements, etc.), and the development of the interaction between users and the application (positive reinforcements, penalties).

The Project has been developed completely using the platform *Google App Engine* for web application development. Data is stored in a space provided by *Google Data Store*, and is managed by *Objectify* (also from *Google*).

Taking all this into account, and once the development phase was over, we have concluded that the results are satisfactory and will allow to get further progress in this line in order to get *Juegamáticas* available to anyone considering using it as support for mathematical learning.

Keywords

Learning, reinforcement, gamification, education, tutor, student, Down Syndrome, evaluation system, Google App Engine, datastore, Objectify, asociation exercises, operation exercises.

Agradecimientos

Cristina y Bea, del Colegio María Corredentora.

A Pilar por desconfiar y tirar.

A Celia por confiar y empujar.

A mi familia.

A Margarita.

INDICE DE CONTENIDOS

INDICE DE FIGURAS

INDICE DE TABLAS

1 Introducción

1.1 Motivación

Antes de finalizar la carrera, todos los estudiantes de Grado de Ingeniería Informática (GII) la Escuela Politécnica Superior (EPS) de la Universidad Autónoma de Madrid (UAM), deben desarrollar y defender un Trabajo de Fin de Grado (TFG) como último gran esfuerzo que les permita poder ser Ingenieros. Llegados a este momento, suelen surgir dudas sobre si hacer un proyecto propio o desarrollar uno de los proyectos que oferta la EPS. Este proyecto pertenece al primer grupo, y nace de la inquietud de poder intentar ayudar a estudiantes con discapacidad a aprender matemáticas.

Esta motivación no surge de la nada, sino que tiene razones personales. Mi hermana pequeña tiene Síndrome de Down, un trastorno genético que entre otras características conlleva discapacidad intelectual.

En casa hemos vivido de forma muy próxima cómo se educa este tipo de niños, cómo hay que estimularlos, y cómo cualquier cosa que para nosotros es relativamente sencilla puede conllevar gran esfuerzo y trabajo para cualquiera de estas personas.

La aplicación desarrollada sirve como apoyo y refuerzo al aprendizaje matemático, ya que la metodología que se emplea para enseñar matemáticas a las personas con discapacidad intelectual puede resultarnos muy alejada, y en muchos apartados difiere mucho de cómo se enseñan este tipo de destrezas a estudiantes sin discapacidades cognitivas. Aparte, no existen apenas aplicaciones educativas dirigidas exclusivamente a estudiantes discapacitados.

Pero, ¿Es realmente necesario aplicar tecnología? ¿Estos estudiantes realmente aprenden con este tipo de aplicaciones? No. La idea no es que aprendan, sino que estas aplicaciones les sirvan de refuerzo. Que puedan dedicar unos minutos al día a una actividad casi lúdica, y que dicha actividad les ayude y motive a continuar aprendiendo

1.2 Objetivo

En general, en los sistemas de educación tradicionales a los estudiantes entre 2 y 3 años ya se les enseña a manejar cantidades y a asociarlas con guarismos. Una vez éstos manejan los números con soltura, se empieza con operaciones sencillas (sumas y restas) y, habitualmente, con unos 7 - 8 años los estudiantes ya están preparados para multiplicar [1]. Sin embargo, a la hora de trabajar con estudiantes Síndrome de Down estas metodologías no son útiles, y hay que aplicar técnicas diferentes para que, en un caso bueno, con 18 – 20 años puedan hacer restas con llevada y sencillas multiplicaciones sin que les suponga un esfuerzo extraordinario.

Dada la complejidad que conllevaba una aplicación como la que nos propusimos, específica para estudiantes Síndrome de Down, decidimos que lo más adecuado era contar con asesoría por parte de educadores especializados en este tipo de discapacidad. Por ello nos pusimos en contacto con la Directora del colegio *María Corredentora*, colegio dedicado específicamente a la educación de estudiantes Síndrome de Down [2], y

decidimos que lo más adecuado era desarrollar la aplicación en colaboración directa con una de las educadoras de la institución, para que pudiesen ayudarnos a comprender y plasmar tanto las metodologías que utilizan para enseñar, como las diferentes problemáticas que se pudiesen plantear en torno a estilos, usabilidad y refuerzos (en el Anexo A encontraremos un resumen de las metodologías que se aplican en este centro).

En cuanto a la parte tecnológica, después de documentarnos sobre las diferentes alternativas y cotejar con las educadoras del colegio *María Corredentora*, decidimos desarrollar nuestra aplicación *Juegamáticas* en formato aplicación web porque, además de que hoy en día, todos los jóvenes son capaces de manejarse con este tipo de tecnología casi sin problema, el colegio de referencia, *María Corredentora*, educa a los estudiantes para que interactúen con este tipo de dispositivos desde muy pequeños, ya que a la hora de comunicarse e interaccionar con una máquina, es mucho más intuitivo para ellos utilizar las manos directamente (Figura 1-1).

Figura 1-1: Estudiante *Síndrome de Down* interactuando con una SmartTable

La aplicación debe estar diseñada para servir de apoyo y refuerzo, y seguir la metodología de enseñanza de la disciplina matemática aplicada en el *Colegio María Corredentora*, para estudiantes de entre 6 y 20 años. Aparte debe hacerse especial hincapié en los diseños y la usabilidad de la misma, ya que esta trisomía suele ir acompañada de diferentes dolencias (problemas de vista, psicomotricidad, etc).

En este centro, las clases suelen ser de unos 15 alumnos, y no todos tienen exactamente el mismo perfil, por lo que la aplicación deberá ir ajustándose a cada usuario.

1.3 Organización de la memoria

El resto de esta memoria consta de los siguientes capítulos:

- **Capítulo 2.** Estado del arte: Descripción de algunas aplicaciones existentes relativas al aprendizaje de las matemáticas para estudiantes discapacitados, o cercanas al objetivo de este trabajo.
- **Capítulo 3.** Análisis y diseño: Descripción de requisitos funcionales y no funcionales,
- **Capítulo 4.** Arquitectura del sistema: Descripción de la arquitectura de la aplicación y las tecnologías aplicadas.
- **Capítulo 5.** Funcionamiento de la aplicación: Descripción del comportamiento del sistema para los diferentes roles de usuario que tendrán acceso a ella.
- **Capítulo 6**. Pruebas y resultados: Descripción de las diferentes pruebas a las que se somete la aplicación, y los resultados de las mismas.
- **Capítulo 7.** Conclusiones y continuación del proyecto: Desarrollo de las diferentes conclusiones y resultados obtenidos, y de los siguientes pasos a seguir para continuar con el proyecto.

2 Estado del arte

2.1 Aplicaciones actuales

En la actualidad, el desarrollo de aplicaciones educativas está viviendo una época dorada debido a la proliferación de dispositivos portátiles. Existen muchas aplicaciones dedicadas al aprendizaje matemático para niños (buscando "matemáticas niños" en el *Play Store* de *Google España*, nos devuelve más de 200 resultados), pero muchas menos son las dedicadas específicamente a estudiantes discapacitados. A continuación describimos algunas de ellas, concretamente *Series 1* [3], *Picaa* [4] *y Eureka* [5].

2.1.1 Series 1

Esta aplicación (para *iOS* y *Android*) está dirigida a niños a partir de tres años con el objetivo de que aprendan a ordenar objetos según forma, color o tamaño, entre otras características, además de ayudar a desarrollar conceptos matemáticos primarios, como el de "cantidad". El juego consta de varios tableros, y cada uno contiene cinco piezas que conforman una serie que los pequeños han de ordenar según la combinación pedida.

Aunque *Series* 1 no está dirigida específicamente para estudiantes con *Síndrome de Down*, es una de las aplicaciones más utilizadas por este tipo de estudiantes. En la siguiente figura se muestra una imagen promocional de *Series 1*.

Figura 2-1: Imagen promocional *Series 1*

2.1.2 Picaa

Picaa es una aplicación (en *iOS*) desarrollada específicamente para niños autistas y con *Síndrome de Down*. Tiene como objetivo mejorar competencias básicas de niños y adolescentes, e incluye actividades de matemáticas, lenguaje, conocimiento del entorno, así como autonomía y habilidades sociales. Apta para pequeños y jóvenes con discapacidad

cognitiva, visual o auditiva, está diseñada para ser utilizada en el aula como herramienta de apoyo. La siguiente figura muestra dos capturas de la aplicación:

Figura 2-2: Imagen promocional *Picaa*

2.1.3 Eureka

Esta aplicación, diseñada para iPad, surge de la necesidad de dar respuesta al desarrollo de las competencias matemáticas básicas de alumnos con necesidades educativas especiales, y está dirigida a estudiantes de Educación Infantil y primeros cursos de Educación Primaria [6]. *Eureka* está promovida por la *Fundación Garrigou* en colaboración con el *Colegio María Corredentora*. Fue presentada el 31 de Mayo de 2016. En la siguiente figura, podemos ver una de las imágenes promocionales de la aplicación:

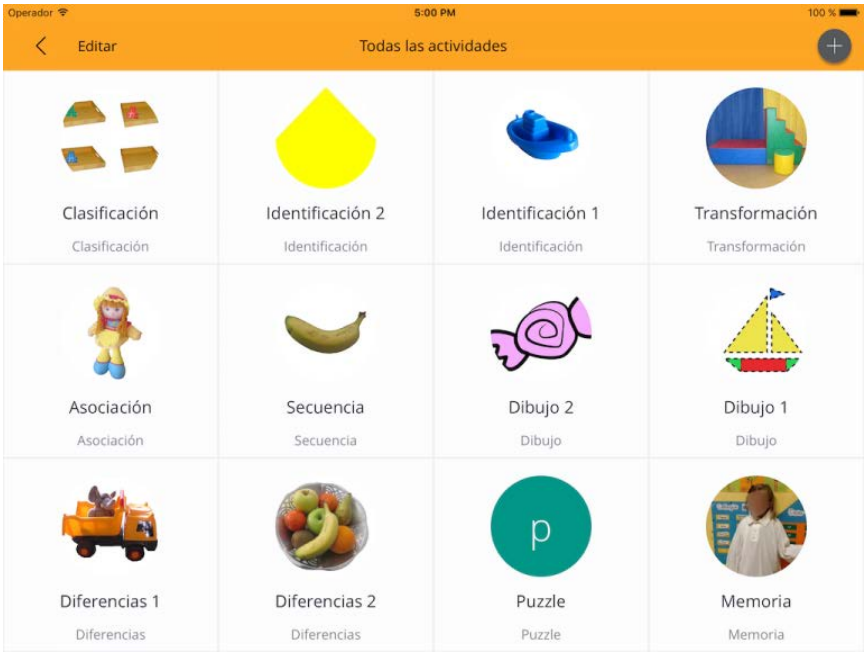

Figura 2-3: Imagen promocional Eureka

2.2 Conclusiones

Después de observar y estudiar las diferentes alternativas, hemos llegado a las siguientes conclusiones:

- Por una parte, las aplicaciones suelen estar compuestas por problemas de tipo puzles, posiciones en el espacio y de orden por tamaño, que son las bases de las matemáticas (sobre todo el orden y el espacio).
- Respecto al tipo de estudiantes al que se dirigen, se trata habitualmente de de un grupo de estudiantes muy acotado en edad: en todos los casos se trata de niños en educación infantil o primaria.
- En cuanto al tipo de dispositivos en los que se encuentran disponibles, únicamente funcionan sobre *IOS* y *Android*.

Como rasgos diferenciadores, nuestra aplicación se centra en problemas más matemáticos puramente hablando (más números, más cantidades, etc.), aunque eso no significa que dejemos de lado algunos temas como tamaño o posición. La idea es aplicar la metodología del *Colegio María Corredentora* de manera directa. Aparte, hemos intentando que la temática y dificultad de los ejercicios acaparase un rango bastante más amplio de edad (entre 7 y 20 años) y hemos hecho especial hincapié en que la aplicación funcionase sobre cualquier dispositivo moderno que tenga un navegador actualizado.

3 Diseño

3.1 Análisis

Para el análisis de requisitos trabajamos directamente con una educadora del *Colegio María Corredentora*. La idea ha sido ir haciendo esquemas, maquetas y prototipos a lo largo del desarrollo y trabajar sobre ellos (basándonos en una metodología espiral basada en prototipos, aunque sin hacer los planes de desarrollo y las pruebas en todas las iteraciones). Según se fueron mostrando los diferentes prototipos, fueron apareciendo requisitos y modificaciones fuera de la primera etapa de análisis de requisitos.

Entre las reuniones e intercambios de correos, podríamos diferenciar 5 etapas:

- Primera etapa: El objetivo de esta etapa era determinar el ámbito de aplicación del trabajo. Inicialmente se había pensado en que la aplicación tuviese ejercicios matemáticos, lingüísticos y de tareas sociales y del día a día. Tras las primeras conversaciones decidimos que la mejor opción era centrarnos en el aprendizaje matemático.
- Segunda etapa: Durante esta etapa estudiamos diferentes temas, tales como la eficacia del juego en el aprendizaje o como método de apoyo. También empezamos a elegir o desechar algunos de los principios en los que basar la aplicación (diversión, frustración, refuerzos positivos, economía de tokens, auto-aprendizaje, etc.).
- Tercera etapa: En esta etapa nos centramos en esbozar algunos de los primeros ejercicios matemáticos que debía incluir la aplicación.
- Cuarta etapa: Aquí fuimos concretando cómo debía ser la evolución de la aplicación para los estudiantes (orden de los tipos de ejercicios, los diferentes niveles dentro de cada tipo de ejercicio, cuándo subir o bajar de nivel, etc.).
- Quinta y última etapa: Por último, nos centramos en el rol Tutor, y en temas de estilos y usabilidad específica para los estudiantes a los que va dirigida la aplicación.

3.1.1 Requisitos Funcionales

El resumen de los requisitos más importantes sería el siguiente:

- RF-1: Acceso sencillo a la aplicación
- RF-2: Aplicación colorida, amigable y con márgenes.
- RF-3: Sistema de respuesta tipo Test mediante botones en pantalla
- RF-3: Botones de respuesta grandes y bien diferenciados
- RF-4: Las preguntas y los resultados tienen que estar compuestos tanto por números que se vean claramente, como por figuras que se diferencien con facilidad (no muy próximas entre ellas)
- RF-5: Posibilidad de volver a responder una pregunta en caso de fallo.
- RF-6: Descripción Escrita y Sonora de los ejercicios
- RF-7: Sonidos de acierto/fallo
- RF-8: Sistema de refuerzo positivo (colores, sonidos...)
- RF-9: No debe valorar tiempos de respuesta (no se debe contabilizar el tiempo que tarda un usuario en marcar una respuesta).

- RF-10: Existirán dos tipos generales de ejercicios: Asociación y Operación,
- RF-11: Seguir una escala de avance de tipos de ejercicio (podrá ser ampliable):
    1. Asociación de figuras iguales.
    2. Asociación de guarismos a figuras.
    3. Asociación de figuras a guarismos.
    4. Parejas fundamentales (figuras y guarismos).
    5. Parejas fundamentales (guarismos).
    6. Operaciones con guarismos.
- RF-12: Internamente, dentro de cada tipo de ejercicio existirán diferentes niveles. A mayor nivel, mayor complejidad. Estos niveles definirán los rangos sobre los que se generarán, aleatoriamente, los ejercicios.
- RF-13: Existencia de 2 tipos de roles: Alumno y Tutor.
- RF-14: Posibilidad de acceso a algunos resultados de uno o varios estudiantes por parte de un maestro o familiar (rol Tutor).
- RF-15: Posibilidad de limitar el tipo de ejercicio máximo que un usuario concreto puede alcanzar (rol Tutor),

3.1.2 Requisitos no funcionales.

Los principales requisitos no funcionales son:

- RNF-1: El acceso a la aplicación debe de poder guardarse, es decir, que exista la posibilidad de que un usuario en un dispositivo únicamente tenga que hacer *login* la primera vez.
- RNF-2: Interfaz intuitiva
- RNF-3: El sistema debe soportar varios usuarios diferentes concurrentemente (unos 20).
- RNF-4: La aplicación debe poder ejecutarse en diferentes navegadores modernos, sobre diferentes sistemas operativos.
- RNF-5: Aplicación *Responsive*, para tablets y smartphones.
- RNF-6: Persistencia: Deberán poder recuperarse los resultados de los usuarios (Alumnos).
- RNF-7: El sistema para avanzar entre niveles y tipos de ejercicio debe ser justo, transparente y que no conlleve frustración (es decir, el Alumno no debe notar que el nivel dentro del tipo de ejercicio sube bruscamente, y el sistema tiene que evaluar qué hacer si un Alumno se queda atascado).
- RNF-8: Un usuario con rol Tutor podrá tener asignados uno o más usuarios con rol Alumno (por ejemplo: un profesor puede tener visibilidad sobre sus alumnos).
- RNF-9: Un usuario con Rol Alumno, podrá tener asignados cero o más usuarios con rol Tutor (por ejemplo: un alumno es visible para sus padres y profesores).

3.2 Diseño

Dado que ya habíamos reunido algunos de los requisitos más importantes de la aplicación, y la aplicación ya tenía iba adquiriendo forma sobre el papel, comenzamos la etapa de diseño, en la que mostramos cómo fuimos construyendo la aplicación (en el Anexo B encontraremos un resumen del modelado de los elementos de la aplicación).

3.2.1 Usuarios: ¿Qué es lo que puede hacer cada rol?

En el RF-12 hablábamos de dos roles diferentes: *Alumno* y *Tutor*. Estos son los roles que se modelaron para la aplicación, pero además hizo falta otro para gestionar la aplicación en sí: *Administrador.*

1. **Administrador**: Es un rol por encima de lo que es la aplicación y los usuarios. Sus labores principalmente serán:
    - Gestión de la Base de Datos.
    - Decidir quién puede ser o no Tutor.
    - Asignar Alumnos a Tutores.
    - Dar soporte a los usuarios.

2. **Alumno**: Es el rol más general de la aplicación. Es el rol que utilizarán los estudiantes o usuarios normales. Principalmente podrá:
    - Resolver ejercicios.

3. **Tutor**: Este rol es el que se utilizará para gestionar uno o varios alumnos. Está pensado para padres y profesores. La primera vez que entre en la aplicación, solicitará que se le asigne el rol de Tutor al Administrador. Dentro de la aplicación, podrá hacer lo siguiente:
    - Consultar resultados de un Alumno (queda pendiente qué información se le mostrará exactamente, y en qué formato)
    - Asignar un tipo de ejercicio máximo a un Alumno

3.2.2 Ejercicios: Tipos de ejercicios.

Todos los ejercicios siguen la misma dinámica: Se plantea una pregunta o problema, y se ofrecen 3 posibles soluciones, entre las que el Alumno puede elegir (formato test). Si el alumno falla una pregunta, se le dará la opción de volver a responderla. Si la acierta, automáticamente le aparece una nueva pregunta. Todos los resultados son almacenados.

El sistema de ejercicios consta de dos tipos generales de ejercicios: Asociación y Operación.

- **Asociación**: La pregunta muestra una figura (puede ser un conjunto de figuras o un guarismo), y el Alumno tiene que elegir la solución que represente esa figura.
- **Operación**: La pregunta muestra una operación, y el Alumno tiene que elegir la solución correcta a esa operación.

Pese a que los tipos generales son bastante simples, serán aplicados para los tipos de ejercicios más particulares. Son con este tipo de ejercicios, con los que conseguimos desarrollar la línea de aprendizaje aplicada en el *Colegio María Corredentora:*

- **Tipo 0**: *Asociación del mismo número de figuras iguales*. Ejercicio de tipo Asociación. Es útil también para que el usuario entienda la dinámica de la aplicación.

- **Tipo 1**: *Asociación del mismo número de figuras distintas*. Ejercicio de Asociación. El usuario tiene que asociar elementos con el mismo número de figuras, teniendo estas figuras diferentes atributos (color, tamaño y posición).

- **Tipo 2**: *Asociación de guarismos a número de figuras*. Ejercicio de Asociación. En este caso, el sistema enseña al usuario un guarismo, y el usuario tiene que elegir cuál de las figuras posibles, tiene el número de elementos correcto.

- **Tipo 3**: *Asociación de número de figuras a guarismos*. Ejercicio de Asociación. Como el ejercicio anterior pero al revés. El sistema muestra una cantidad de elementos y el usuario tendrá que elegir el guarismo correcto.

- **Tipo 4**: *Parejas fundamentales (figuras + guarismos)*. Ejercicio de Operación. El sistema presentará una suma, en el que el guarismo viene acompañado de una imagen con el número de elementos. Todas las sumas que aparecen en este ejercicio, pertenecen al grupo de parejas fundamentales utilizados para representar los números del 2 al 9.

- **Tipo 5**: *Parejas fundamentales (guarismos)*. Ejercicio de Operación. Como el ejercicio anterior, pero únicamente con guarismos.

- **Tipo 6**: *Operaciones con guarismos*. Ejercicio de Operación. Sumas con guarismos.

En el Anexo C encontraremos algunas capturas de estos ejercicios, de una versión avanzada de la aplicación.

3.2.3 Ejercicios: Niveles y sistema de evaluación.

Dentro de cada tipo de ejercicio existen 6 diferentes niveles que marcarán el rango de valores a los que se enfrentará el usuario, y que serán transparentes para el mismo. Para decidir si el usuario debe subir o bajar de nivel se seguirán las siguientes reglas:

- Si el Alumno tiene un porcentaje de acierto mayor al 80% en un nivel, pasa al siguiente. Si está en el nivel máximo dentro de un tipo de ejercicio, pasará al siguiente tipo de ejercicio.
- Si el Alumno tiene un porcentaje de acierto menor al 50.1%, bajará de nivel. Si está en el nivel mínimo dentro de un tipo de ejercicio, bajará de tipo de ejercicio.

En ambos casos, si cambia de tipo de ejercicio, se reinician las estadísticas que tenía sobre ese tipo de ejercicio. Esto es así para que si baja de tipo de ejercicio, empiece desde el nivel más sencillo dentro de ese tipo de ejercicio, y evitar que se frustre quedándose atascado en un punto. En el Anexo D encontraremos la implementación de este algoritmo en java.

- Dentro de cada tipo de ejercicio, las preguntas se irán generando con valores aleatorios dentro del rango del nivel:
- Los ejercicios de Asociación (Tipo: 0, 1, 2, 3) generarán aleatoriamente tanto la pregunta como las respuestas. Siempre dentro de los siguientes rangos:

	Pregunta	Respuesta
Nivel 0	[1..3]	[1..3]
Nivel 1	[1..3]	[1..9]
Nivel 2	[4..6]	[4..6]
Nivel 3	[4..6]	[1..9]
Nivel 4	[7..9]	[7..9]
Nivel 5	[7..9]	[1..9]

Tabla 3-1 : Rangos ejercicios de tipo Asociación

- Los ejercicios de operaciones con Parejas Fundamentales (Tipo 4 y 5): Inicialmente generará el resultado correcto dentro del rango del nivel, y con el resultado ya generado, se creará la operación asociada a esa pareja fundamental:

	Respuesta
Nivel 0	[2..4]
Nivel 1	[2..5]
Nivel 2	[5..7]
Nivel 3	[2..9]
Nivel 4	[5..9]
Nivel 5	[8, 9]

Número	P.F.
2	1+1
3	1+2
4	2+2
5	2+3
6	3+3
7	4+3
8	4+4
9	4+5

Tabla 3-2 : Rangos ejercicios Parejas Fundamentales

- Por último, los ejercicios de operaciones con guarismos general (Tipo 6), generará la pregunta aleatoriamente dentro del rango del nivel:

	Pregunta
Nivel 0	[0..3]+[0..3]
Nivel 1	[4..6]+[0..3]
Nivel 2	[4..6]+[0..6]
Nivel 3	[4..6]+[4..6]
Nivel 4	[7..9]+[0..9]
Nivel 5	[7..9]+[7..9]

Tabla 3-3 : Rangos ejercicios de operaciones con guarismos

4 Arquitectura

4.1 Arquitectura de la aplicación

Para implementar la infraestructura de nuestra aplicación utilizamos una arquitectura de tipo cliente-servidor, y a nivel de aplicación aplicaremos el patrón de diseño Modelo Vista Presentador.

4.1.1 Modelo cliente servidor

Dado que estamos desarrollando una aplicación web, el **Modelo Cliente Servidor** consideramos que era el más acertado para nuestro propósito (figura 4.1).

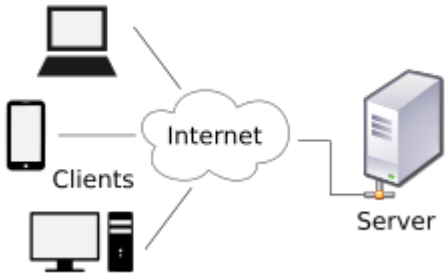

Figura 4-1: Diagrama Modelo Cliente Servidor [7]

La idea de este modelo es que los Clientes (navegadores web de ordenadores, Tablets o Smarphones) se conectan a través de internet contra un Servidor, el cual envía a los clientes los recursos necesarios para la ejecución de la aplicación [8].

Al aplicar este modelo, conseguiremos que la mayor parte de la lógica esté fuera de la parte cliente, de manera que no haga falta que sea un cliente muy potente, ni por supuesto el cliente tiene acceso a la parte interna de la aplicación.

4.1.2 Modelo Vista Presentador

Como en casi todas las aplicaciones, hay que pensar qué patrón de diseño es el que debemos aplicar. En nuestro caso, y dada la naturaleza de la aplicación, debemos buscar un modelo que permita separar la lógica de la aplicación de la interfaz de usuario (como mencionamos en la sección anterior, en el Anexo B se encuentra un resumen del modelado de los elementos de la aplicación).

Figura 4-2: Diagrama MVP [9]

En nuestro caso nos hemos decantado por el **Modelo Vista Presentador** (**MVP**). Este patrón de diseño tiene 3 actores, como podemos ver en la figura 4-2:

- **Vista**: Actor encargado de mostrar la información al usuario (Interfaz de aplicación).
- **Modelo**: Actor encargado de implementar la lógica de la aplicación. Desconoce cómo se va a presentar la información.
- **Presentador**: Es el actor encargado de comunicar a los otros dos, manteniendo la independencia entre ellos.

Para nuestro caso, pongamos por ejemplo la representación de la clase **Ejercicio**:

- **Ejercicio_modelo**: Es el que se ocupa de recoger los datos y fabricar un Ejercicio nuevo. Una vez recoge el resultado (fallo o acierto) del ejercicio anterior, calcula si se debe pasar o bajar de nivel y generará un nuevo ejercicio en consecuencia al nuevo nivel o tipo de ejercicio (si el usuario ha pasado de nivel, o ha bajado de nivel) tal y como se explica en el punto 3.2.2 y 3.2.3 de este documento.
- **Ejercicio_vista**: Es el encargado de recoger los datos de la parte modelo (sin tener ni idea de cómo o por qué se han generado estos), y representarlos para el usuario. Dependiendo del tipo de ejercicio que venga, ejecutará un tipo de representación u otra.
- **Ejercicio_presentador**: Es el encargado de comunicar a los otros dos actores. Si en la *Ejercicio*_vista el usuario resuelve correctamente el ejercicio, el presentador le comunica los resultados al *Ejercicio*_modelo y le pide uno nuevo, para devolvérselo al *Ejercicio_vista*.

4.2 Tecnologías

4.2.1 Cloud Computing: Google App Engine

El *Cloud Computing* es un paradigma que permite ofrecer servicios computacionales a través de la red [10]. Hemos elegido esta opción para nuestro desarrollo dado que se trata una aplicación web, y que pretendemos que sea una aplicación con “terminales tontos” y distribuida.

Para ello, hemos utilizado la tecnología de **Google App Engine** (**GAE**), cuyo logo se muestra en la siguiente figura**.**

Figura 4-3 : Google App Engine Logo

GAE es la solución de Google para el desarrollo de aplicaciones *Cloud*, la cual proporciona un conjunto de herramientas para poder crear aplicaciones web y desplegarla tanto en local, como en servidores propios de Google o en servidores externos.

Existen los dos modelos típicos de negocio para un servicio de estas características: gratuito y de pago. Para el desarrollo de este proyecto, hemos elegido la opción gratuita, que pese a estar limitada, nos ha servido de manera satisfactoria.

Las características más importantes de este servicio son [11]:

- **Lenguajes**: Soporta varios diferentes como Java, Python y PHP. En nuestro caso, desarrollamos la aplicación en **Java**.
- **Gratuito:** eliminamos los costes a corto plazo.
- **Documentación** de fácil acceso en los entornos de Google.
- **Cloud:** Acceso a la aplicación desde cualquier sitio, en cualquier momento, y sobre cualquier navegador actualizado.
- **Conectividad:** Dado que las aplicaciones se alojan de manera distribuida en servidores de Google, las probabilidades de una caída de servicio son muy bajas.
- **Privacidad:** Los datos de usuarios se almacenan en los mismos servidores de Google. En nuestro caso, los usuarios necesitarán una cuenta Google para poder acceder a la aplicación (en el *Colegio María Corredentora* trabajan con este tipo de cuentas), con ello evitaremos el tener que guardar información sensible de los mismos (por ejemplo, contraseñas), diferenciando a cada uno únicamente por su dirección de correo.
- **Seguridad:** La seguridad de una aplicación web depende de la implantación de medidas por parte de Google en GAE y sus servidores, y de cómo desarrollemos la aplicación. Para esta última parte, GAE tiene un analizador de código estático que permite al administrador analizar sus aplicaciones siempre que quiera.
- **Consola:** Posee una potente y sencilla consola de administración web desde la que podemos controlar la aplicación en todo momento (levantar y parar estancias, controlar almacenamiento, análisis de seguridad, etc.). En la Figura 4-4 podemos ver una captura actual de la consola web de GAE.

Por otra parte, algunos de los **límites** de la versión gratuita:

- 5GB de almacenamiento en la nube.
- 1GB para código y logs.
- 657 000 llamadas a la API diarias.
- 50 000 operaciones de lectura y escritura diarias.
- 22 MB de comunicación por minuto.
- 100 correos al día.

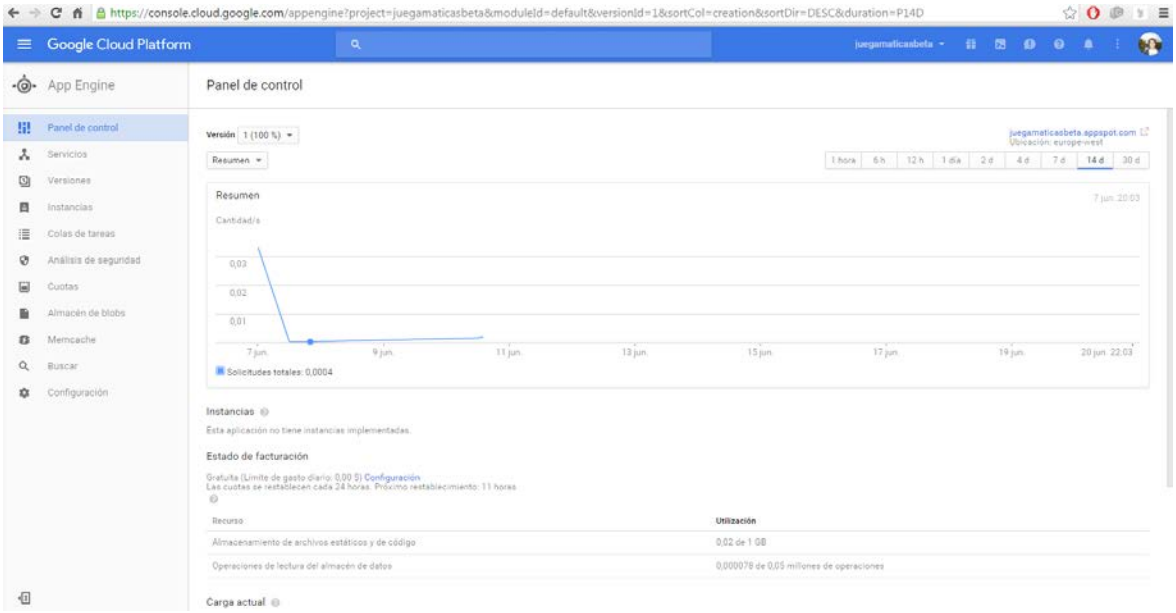

Figura 4-4: Consola GAE

4.2.2 Persistencia de datos: Objectify

Para la persistencia de datos hemos elegido Objectify, una interfaz (API) para el motor de almacenamiento de aplicaciones de Google.

Algunas de sus características y objetivos más importantes son:

- Proporcionar servicio de tipo NoSQL.
- Mantener las características nativas de la Datastore (almacenamiento de datos GAE) de forma intuitiva.
- Modelar estructuras de datos polimórficas, sofisticadas y de tipado fuerte.
- Permitir lógica transaccional tipo EJB modular.
- Aumentar el rendimiento y reducir los costes a través de una caché inteligente.
- Permitir grandes migraciones de esquema “in situ” con cero tiempo inactivo.
- Coexistir sin problemas con otras herramientas del almacén de datos.

Objectify soporta transacciones y, entre otros, incluye los siguientes conceptos:

- **Entidad**: Objeto Java persistente creado mediante una clase POJO (Plain Old Java Object). Entidades del mismo tipo no tienen por qué tener los mismos atributos, aunque sí un identificador único.**Índices**: Es un concepto muy parecido a los índices de las bases de datos (BBDD) relacionales clásicas. En el caso de Objectify, los índices de las entidades son el identificador único de cada una. Con ello conseguimos eficiencia en las consultas. Existen algunos tipos que no pueden ser indexados (*Text*, *Strig* de más de 500 caracteres), y podremos definir índices complejos en un archivo externo (*index.yaml*).
- **Querys**: Al igual que en las BBDD relacionales, se pueden hacer consultas sobre un tipo de entidad en la que podremos aplicar filtros sobre campos y criterios de ordenación

- **Callbacks**: Un callback nos permite ejecutar el código en varios puntos en el proceso de persistencia. Algunos ejemplos son:
  - **PrePut**: Callback invocado antes de que cualquier entidad de la clase especificada se añada.
  - **PostPut:** Callback será invocado después de que cualquier entidad de la clase especificada se añada.
  - **PreDelete:** Callback será invocado antes de que cualquier entidad de la clase especificada se borre.
  - **PreGet:** Callback será invocado antes de obtener entidades de la clase especificada.
  - **PreQuery:** Callback será invocado después de obtener entidades de la clase especificada.

4.2.3 Google Web Toolkit

Google Web Toolkit (GWT) es una herramienta del ecosistema GAE pensada para desarrollar aplicaciones Java en la web.

Cara al desarrollador, la idea es bastante simple: Desarrolla una aplicación Java, que GWT se ocupa en transformar las partes que quieras (por ejemplo, todo lo que esté en la parte cliente) en JavaScript y HTML.

Además, ofrece bastantes ventajas:

- Compatibilidad con navegadores modernos (*Google Chrome, Chromium, Firefox, Safari* 5 y 6, *Opera* a partir de la versión 30, *Internet Explorer* a partir de la versión 8).
- No hace falta tener nociones profundas de diseño y desarrollo web, y mediante ficheros XML podremos configurar casi todo (por ejemplo: qué elementos se transforman en JavaScript)
- Implementación rápida y sencilla de MVP.
- Analiza y Optimiza el código.
- Comunicación mediante llamadas a procedimientos remotos (*Remote Procedure Calls* o **RPC**).
- Integración con *JUnit* (tests unitarios).
- Integración con *BootStrap*

Pero también presenta ciertos inconvenientes:

- Pese a ser compatible con el motor de *Internet Explorer*, el navegador de *Microsoft* presenta varias carencias a la hora de reproducir una aplicación GWT (aunque esto en realidad es culpa del navegador).
- Sólo está implementado para desarrollar con Java y/o JavaScript embebido.
- Hay que tener mucho cuidado con las configuraciones, pueden llevar a que la aplicación no se comporte igual en local que en remoto, o a que sin querer intentes transformar a JavaScript parte del código servidor (lo que es incompatible con Objectify).
- Hay que compilar todo el código cuando hacemos pequeños cambios.

GWT divide el directorio principal en tres carpetas diferentes: *client*, *server* y *shared*. La carpeta *client* contiene el código que va a ser ejecutado en nuestro navegador, el directorio *server* contiene el código que sólo correrá en el servidor y el directorio *shared* contiene código al que el cliente y el servidor acceden de forma compartida.

Una vez compilado el proyecto, se crea un directorio *war* donde se guarda todo el código HTML, **Cascading Style Sheets (CSS)**, JavaScript y archivos *jar* generados. Crea también un archivo XML, como el que vemos en la Figura 4-5, en el que se especifican los módulos de nuestro proyecto y el punto de entrada de la aplicación.

```
<!-- Default page to serve -->
<welcome-file-list>
    <welcome-file>Juegamaticas.html</welcome-file>
</welcome-file-list>

<filter>
    <filter-name>ObjectifyFilter</filter-name>
    <filter-class>com.googlecode.objectify.ObjectifyFilter</filter-class>
</filter>
<filter-mapping>
    <filter-name>ObjectifyFilter</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>

<!-- Servlets -->
<servlet>
    <servlet-name>loginService</servlet-name>
    <servlet-class>com.juegamaticas.server.LoginServiceImpl</servlet-class>
</servlet>
```

Figura 4-5 : Ejemplo Web.xml

4.2.4 RPC

Las llamadas a procedimientos remotos son programas que se utilizan para ejecutar código en otra máquina sin preocuparnos de la comunicación a bajo nivel.

GWT está preparado para integrarse perfectamente con esta opción de comunicación, por ello ha sido la que hemos elegido para que la "parte cliente" y la "parte servidor" se hablen, y lo hagan de forma asíncrona.

A la hora de utilizar RPC con GWT hay que centrarse en los 3 elementos que intervienen en la llamada a procedimientos remotos [12]:

- El servicio que se ejecuta en el servidor (el método al que llamaremos).
- El código cliente que invoca al servicio.
- Los objetos de datos Java que son comunicados entre el cliente y el servidor.

Por ello, tendremos que implementar:

- Una interfaz que represente al servicio y que extenderá de *RemoteService*.
- Una clase que extienda de *RemoteServiceServlet*. Esta clase implementará también la interfaz definida en el punto anterior.
- Por último, una interfaz asíncrona que será llamada desde el lado del cliente.

En nuestro caso, hemos tenido que implementar servicios para el Login (ver imagen anterior), Ejercicios y Usuarios, de manera que la comunicación (sobre todo para la llamada a métodos) funcionaba sobre RPC.

4.2.5 Otras tecnologías

Aparte de las nombradas anteriormente, se han utilizado diferentes tecnologías para la implementación de la aplicación:

4.2.5.1 TTSReader

Se trata de una aplicación (web, y también disponible para *Android* e *iOS*) gratuita bastante elaborada, que entre otras funcionalidades tiene la posibilidad de leer un texto [13].

Dentro de la vista de un ejercicio de *Juegamáticas*, aparece un botón (Figura 4-6) que al pulsarlo reproduce un audio con la descripción sonora del enunciado del tipo de ejercicio:

Figura 4-6 : Botón de reproducción sonora del enunciado.

Esta reproducción del enunciado, tiene que ser espaciada y con buena dicción. Por ello, nos decantamos con esta tecnología, que ofrece diferentes velocidades a la hora de leer un texto, diferentes acentuaciones e idiomas, y una voz clara, natural y humana.

4.2.5.2 Audacity

Audacity (logo en Figura 4-7) es un editor de audio gratuito y libre. Es el complemento a *TTSReader*: Con uno generamos los audios para la aplicación, y con el otro los cortamos y grabamos para generar los archivos que se incluirán como estáticos de la aplicación.

Figura 4-7: Logo Audacity

5 Funcionamiento de la aplicación

Una vez implementadas las funcionalidades principales de la aplicación, en esta sección se detalla cómo funciona actualmente *Juegamáticas* para cada uno de los roles.

5.1 Usuario Administrador

Este usuario no es un usuario como tal de la aplicación, con esto quiero decir que no es un usuario que vaya a evaluar o ser evaluado, ni siquiera aparece en base de datos, ni está modelado dentro de la aplicación. El Administrador es el que se encarga, esencialmente, del mantenimiento, soporte y mejora de la aplicación. Sus labores más importantes son:

- Recibir dudas o peticiones de los usuarios de la aplicación.
- Asignar tutores: Qué usuarios de la aplicación son tutores y quiénes son sus tutelados.
- Realizar cambios en base de datos: Puede ocurrir que exista cierto interés en eliminar o modificar parte de la información almacenada, y este usuario es el único que debe poder hacer estos cambios. La idea no es que modifique las puntuaciones o haga algo para perjudicar o favorecer a un usuario, si no que hoy por hoy, y dado que aún estamos haciendo diferentes pruebas, poder hacer los cambios que necesitemos "en caliente", es algo que ayuda y agiliza notablemente.

A pesar de que no existe el objeto que implemente este usuario dentro de la aplicación, me parecía importante incluirlo en la memoria ya que sí que es un rol importante dentro de este proyecto, y en un futuro será un rol que de alguna forma aparecerá modelado dentro de la aplicación, aunque aún no sabemos si creando un objeto usuario de este tipo, o delegando parte de sus métodos sobre el usuario Administrador.

5.2 Usuario Tutor

El usuario Tutor es el encargado de visualizar y valorar los resultados de algunos de los usuarios con rol Alumno. Esto es, un tutor tiene visibilidad sobre varios alumnos asignados a él y puede consultar algunos de sus resultados, o modificar cuál es el tipo de ejercicio máximo (que no nivel) al que puede acceder el alumno

5.2.1 Login

En el primer acceso a la aplicación, aparece un mensaje que avisa de que es necesaria una cuenta Google para acceder a la aplicación, Figura 5-1 a continuación:

Figura 5-1 : Mensaje previo al Login

Una vez el usuario presiona el botón “Entrar”, aparece la pantalla de bienvenida de google, en la que se requiere que el usuario introduzca sus credenciales. Otra opción es que si el navegador tiene constancia de una cuenta activa de Google, Figura 5-2, permite acceder con un click:

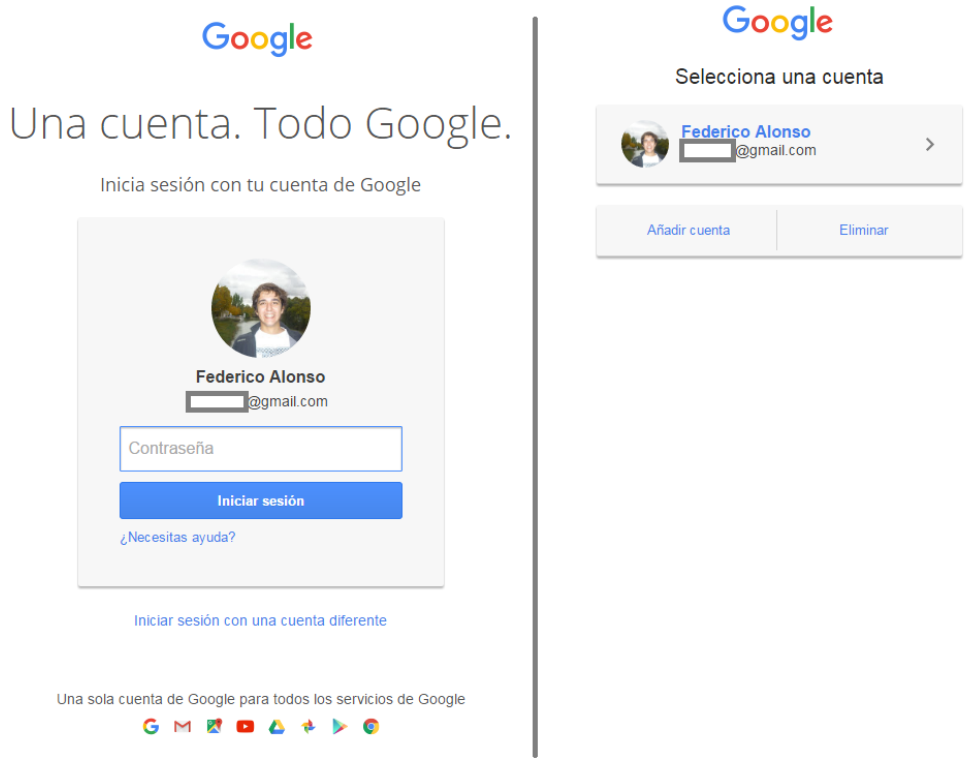

Figura 5-2 : Alternativas de acceso a la aplicación

Una vez un usuario hace su primer registro en la aplicación, el rol que se le asigna automáticamente es el de Alumno. Pero llegados a este punto, puede solicitar ser Tutor y a quién quiere tutelar, mediante un correo a *soportejuegamaticas@gmail.com*. La idea de que se siga este camino para ser tutor viene de intentar tener la interfaz lo más limpia posible para los usuarios de tipo Alumno, es decir, valoramos en un principio la opción de poner un botón para solicitar ser Tutor, pero ese botón debía estar después del login, y ya en una primera prueba vimos que no era buena idea poner un botón de esas características, ya que llevaba a que un usuario con discapacidad lo pulsase y le apareciese un formulario (en el que se pedían entre otras cosas los correos de los usuarios a tutelar) que no tenía nada que ver con la dirección que debía tener para ese usuario la aplicación. Ésta es una solución de contingencia (más adelante, hablaremos sobre el trabajo futuro, en donde trataremos también de este punto.).

5.2.2 Mis alumnos

El punto fuerte del usuario Tutor, es el ser el responsable de uno o más usuarios de rol Alumno. Dentro de la vista *Mis alumnos* el profesor puede buscar a uno de sus alumnos, y una vez lo tiene localizado podrá hacer dos cosas:

- Consultar los resultados del alumno (en qué tipo de ejercicios y nivel ha fallado, ha acertado, etc.), de manera que pueda controlar la evolución del mismo.

- Editar el tipo de ejercicio máximo autorizado para el alumno: Con esto conseguimos que el usuario no se vaya a un tipo de ejercicio que requiera un conocimiento más profundo de la materia, y que le acabe frustrando.

5.3 Usuario Alumno

Como ya hemos explicado con anterioridad, este es el rol principal hacia el que se ha dirigido la aplicación.

El login es necesario hacerlo únicamente la primera vez. Mientras que el usuario utilice el mismo dispositivo, y no le dé a "Salir" de la aplicación, no tendrá que volver a loguearse.

5.3.1 Login

El login es el mismo que para un usuario con rol Tutor.

El que haya una pantalla de login, puede parecer contraproducente para el tipo de usuario al que está dirigido, pero era la mejor alternativa para poder diferenciar a los usuarios (y sus datos) una vez está dentro de la aplicación. Aparte, está pensado que el primer uso sea guiado por algún tutor.

La decisión de utilizar cuentas de Google para el acceso a la aplicación está basada en los siguientes argumentos:

- Todo el proyecto lo hemos hecho sobre un entorno *Googleizado*, en el que casi el completo de las tecnologías pertenecen a esta empresa, y casan perfectamente entre ellas.
- El utilizar cuentas Google facilita el desarrollo, ya que no hay que preocuparse de hacer desarrollo (tanto de *front* como de *back*) de la vista, ni de securizar los campos.
- Por último, el *Colegio María Corredentora* utiliza este tipo de cuentas con sus alumnos, es decir, llegados a una cierta edad todos los alumnos tienen cuenta en Google.

5.3.2 Ejercicios

Casi todos los requisitos funcionales recogidos durante las primeras etapas del proyecto se centran en este punto, por lo que ha sido donde se ha centrado casi todo el esfuerzo de desarrollo.

En cuestión de **usabilidad** y diño, la principal característica es la sencillez con la que se presenta la información. En la Figura 5-3 vemos cómo se han distribuido los elementos para cumplir con los requisitos:

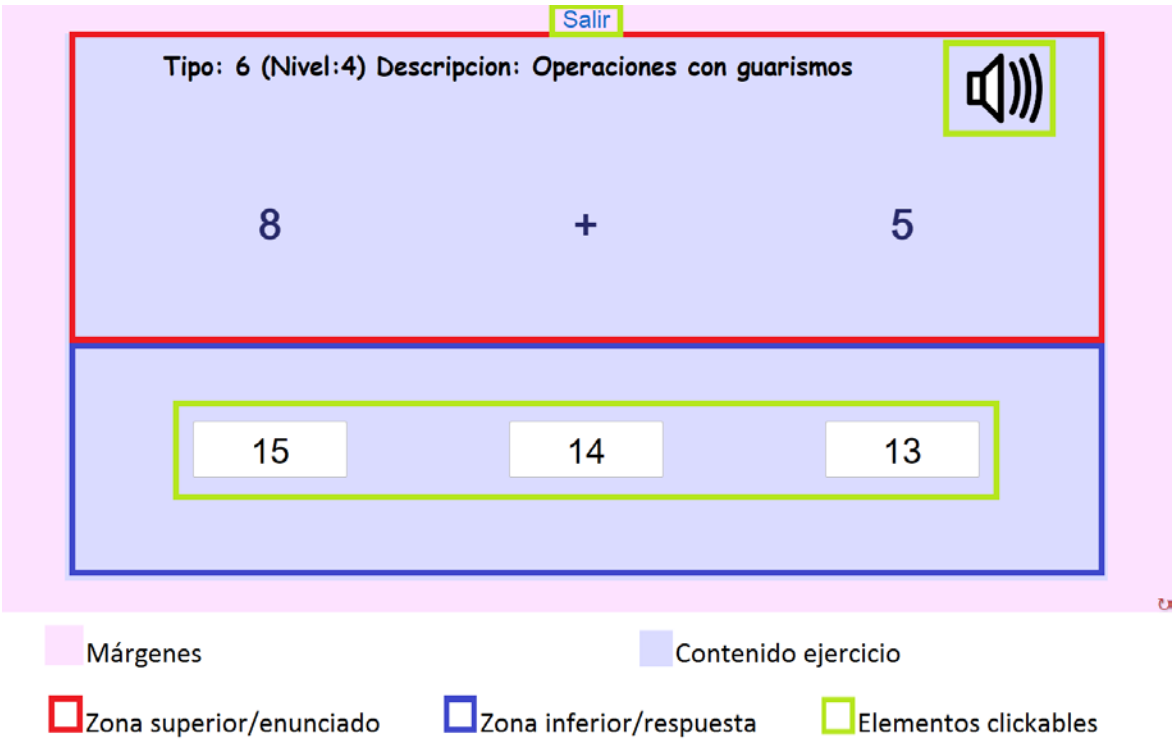

Figura 5-3 : Resumen de los elementos de los ejercicios[1]

- **Márgenes**: Los márgenes de la vista de cualquiera de los ejercicios están pensados para que un usuario pueda correr la aplicación sobre un dispositivo táctil y portátil (una Tablet o Smartphone) agarrándolo cómodamente y sin que corra el riesgo de interactuar con la aplicación sin querer. El único elemento clickable que se ha incluido en estos márgenes es el botón de "Salir", que cierra la sesión del usuario. Este botón se encuentra en el margen superior, y la elección de esta posición es porque raras veces agarran el dispositivo por la parte superior (la aplicación es *responsive*, por lo que si giran el dispositivo, los elementos cambiarán de orientación también, y se reajustarán a la nueva disposición), y porque consideramos que no era buena idea poner el botón de cierre de sesión en una zona donde el usuario pudiese interactuar con él por error.
- **Zona superior/Enunciado**: Dentro de la parte central de la vista de ejercicios (lo que consideramos el ejercicio en sí), en la parte superior encontramos el enunciado del tipo de ejercicio actual: El enunciado escrito (en la captura aparece el tipo, nivel, y una breve descripción, pero como hemos notificado con anterioridad, estas capturas pertenecen a una versión *alpha*, que no se corresponden con las actuales) y el enunciado en formato sonoro al pulsar el botón con la figura de un altavoz (al pulsarlo, reproducirá directamente una grabación con el sonido del enunciado genérico para ese tipo de ejercicio, no dirá "ocho más cinco"). Justo debajo de esa zona encontramos el ejercicio en sí a resolver en ese momento. Estos ejercicios, como explicamos anteriormente, varían según el usuario va evolucionando dentro de la aplicación.

[1] Todas las capturas añadidas en este documento han sido tomadas en una **versión de prueba**, y no representan necesariamente el estado actual o final de la aplicación.

- **Zona inferior/Respuesta**: Dentro de la parte central de la vista de ejercicios, en la parte inferior encontramos 3 elementos clickables que corresponden con las 3 posibles soluciones entre las que puede elegir un usuario. Al igual que en el enunciado, dependiendo de el tipo de ejercicio en el que encuentre el usuario en ese momento, los botones de solución varían su forma (pueden aparecer, figuras del mismo color y forma, guarismos, figuras con color y forma diferentes, etc.). Se ha hecho especial hincapié en que el usuario pueda corregir un ejercicio en caso de que seleccione una respuesta errónea, por lo que si ocurre esto, la aplicación marcará la opción elegida en otro color, pero se mantendrá en el ejercicio (Figura 5-4).

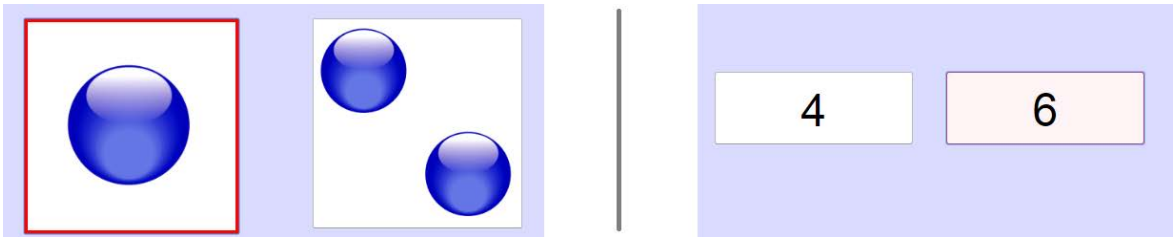

Figura 5-4 : Si el usuario elige una respuesta incorrecta, el botón elegido cambiará de color

Cuando un usuario responde a una pregunta, la aplicación reproducirá un sonido diferente en caso de acierto, de fallo, y cuando acierta y cambia de tipo de ejercicio, de forma que el usuario se sienta animado a continuar con el trabajo. También habíamos puesto un sistema de puntuación (acierto sumaba 100 puntos, fallo restaba 20), pero en la última reunión en el colegio, nos comentaron que no era buena idea mantener ese tipo de refuerzo, ya que estamos reforzando lo aprendido sobre numeración con guarismos y representaciones numéricas pequeñas, a la par que mostrábamos una puntuación con números (guarismos) grandes, lo que podía provocar cierta confusión en el usuario. Actualmente, estamos buscando un sistema de refuerzo alternativo a implementar en la siguiente versión.

Cuando un usuario deja de contestar ejercicios, se deja de grabar información, y cuando retoma el uso con la aplicación, se cargan los últimos valores y estadísticas de manera que continúa en el mismo nivel y tipo de ejercicio que donde lo dejó en su última sesión. Con esto pretendemos explicar que no hace falta que cierre sesión ni nada parecido una vez deja de utilizarla, ya que el guardado de datos y la generación de nuevos ejercicios funciona de forma asíncrona intercambiándose los datos únicamente cuando el usuario pulsa una posible solución.

Juegamáticas ha ido cambiando el diseño y funcionamiento de los ejercicios para ir ajustándose a los requisitos y comentarios de las versiones que se fueron presentando en el *Colegio María Corredentora.*

6 Pruebas y resultados

En este apartado hablaremos de las diferentes pruebas a las que hemos sometido a la aplicación, desde unitarias hasta aceptación. El *testing*, o pruebas efectivas, que se puede hacer a cualquier aplicación tiende a infinito, por lo que nosotros nos centramos en resolver los casos más habituales (pese a que yo era el único desarrollador, y no es bueno que sea el desarrollador el que realice las pruebas técnicas).

6.1 Pruebas

6.1.1 Pruebas Unitarias

Pese a que GAE está integrado con *JUnit*, no hemos empleado ningún *framework* de este tipo para realizar estas pruebas. Hubo sobretodo dos motivos por los cuales tomamos esta decisión:

- El código de la aplicación utiliza métodos bastante simples, y que se pueden verificar haciendo un uso de la aplicación, por lo que los casos extremos los probamos manualmente (utilizando métodos *main*).
- Dado que en este entorno, aunque despliegues la aplicación en local, no se puede depurar (no tiene un entorno o aplicación que cumpla este cometido correctamente) para repasar la aplicación a conciencia lo que hicimos fue utilizar los logs para este cometido.

Por ello, fuimos construyendo y repasando la aplicación poco a poco, y finalmente consideramos que no era necesaria la creación de un módulo de *testing* dirigido a pruebas unitarias.

6.1.2 Pruebas de integración

Podemos considerar que *Juegamáticas* está compuesto por 3 subsistemas: *Frontend*, *Backend* y *BBDD*. Las pruebas de integración que hicimos sobre nuestra aplicación se basaban en la filosofía de "Si terminamos de hacer una funcionalidad (tanto si es nueva, como si es una modificación de lo anterior) en la que intervenga un módulo diferente, tanto de su mismo subsistema como de uno externo, es necesario probar".

Para hacer este tipo de pruebas, primero probábamos únicamente la nueva funcionalidad, y luego, utilizando de base los casos de uso (algunos de los cuales se encuentran en el Anexo E de este documento) que guardasen alguna relación con la nueva funcionalidad, íbamos haciendo pruebas, cada vez más complicadas.

Por ejemplo, tomemos que introducimos la funcionalidad de reproducir sonidos de acierto, fallo y subida de tipo de ejercicio. Un set de pruebas para validar esta nueva funcionalidad podría ser:

- ¿Suena al cambiar de ejercicio?
- ¿Son los sonidos correctos para los casos de acierto, fallo y subida de tipo de ejercicio?

- ¿La aplicación reproduce algún sonido cuando se desciende el tipo de ejercicio?
- Repetir las pruebas en sistemas móviles y otros navegadores.

Ejecutando este tipo de pruebas fue donde encontramos casi todos los errores (No se cargan las estadísticas correctamente desde la base de datos, descender varios niveles en vez de 1, los png de los botones no se redimensionaban bien en pantallas pequeñas, etc).

Pese a que no pertenecen a este grupo exactamente, en el Anexo F encontraremos algunos resultados de las pruebas de integración de la aplicación con diferentes dispositivos.

6.1.3 Pruebas de carga y pruebas de estrés

Este tipo de pruebas no se pueden llevar a cabo correctamente en nuestro entorno dadas las limitaciones de la versión gratuita de GAE (descritas en el punto 4.2.1 de este documento).

Aun conociendo las limitaciones del sistema, hicimos un estudio para comprobar si podía haber problemas a la hora de que 20-30 alumnos hiciesen una sesión de 15 minutos al mismo tiempo.

Para hacer este estudio, supusimos el caso (un poco extremo) con las siguientes reglas:

- Todos los alumnos son nuevos en la aplicación.
- Los 100 ejercicios que resuelven son de tipo 2 (el más pesado).
- La caché funciona un 50% de las veces.
- Las medidas se redondearán un poco al alza.

Cálculo:
Desde el acceso a la URL hasta que carga el primer ejercicio completo (la mitad de estas peticiones son contra el sistema de Google, pero las interpretaremos nuestras):

19 peticiones (650KB)

Petición de ejercicio tipo 2:

La primera: 13 peticiones (120KB)

El resto: 5 peticiones (2 - 40KB). Con los fallos de caché (no encuentra la imagen, el sonido, o cualquier otro estático), supongamos 25 KB.

Por usuario:

1 acceso a la aplicación + 1 primer ejercicio + 99 ejercicios
19p (650KB)+ 13p (120KB) + 99*(5p (25KB))
527 peticiones (2,48 MB) por Usuario

30 Usuarios:

Total = 15810 peticiones (74,4 MB)

Recordemos los límites de los que hablamos en el apartado 4.2.1:

Límite	Comentarios	Resultados	Favorable
5GB de almacenamiento en la nube.		0.00238 GB	OK
1GB para código y logs.		106,16MB	OK
65 000 llamadas a la API diarias.	Supongamos que todas las llamadas implican API	15810	OK
50000 operaciones de lectura y escritura diarias.	Supongamos que todas las llamadas implican R/W	15810	OK
22 MB de comunicación por minuto.	Supongamos que nuestro resultado está calculado para 10 minutos.	7,44	OK
100 correos al día.	No utilizamos este servicio		OK

Tabla 6-1 Comparativa requisitos[2]

Como podemos observar, no habría problema en cumplir los requisitos. Si quisiésemos escalar la aplicación (con las condiciones expuestas), encontraríamos el cuello de botella en la comunicación por minuto, la cual llegaría a su límite al escalar hasta **295%**.

6.1.4 Pruebas de seguridad

A la hora de analizar la seguridad de una aplicación web, debemos hacer los análisis o auditorías de 2 maneras:

- Análisis dinámico: Sirve para encontrar vulnerabilidades en la infraestructura. Para ello lo primero que se suele hacer es un "descubrimiento" mediante alguna herramienta (por ejemplo *NMAP* [14]), y una vez nos devuelve los resultados (en donde cogeremos los activos que puedan ser vulnerables), lanzamos una sonda (por ejemplo un *Nessus* [15]) para escanear esas vulnerabilidades, esperaremos los resultados y actuaremos en consecuencia de los mismos. En nuestro caso, confiamos en que la infraestructura de *Google* sea segura.
- Análisis estático: Se lanza sobre el código de la aplicación, el escáner analiza el código y te dice donde se encuentran los puntos débiles de la aplicación. Existen varias herramientas (*bugScout, FortiFy, Veracode, Checkmarx,* etc. [16]). En nuestro caso, y dado que permanecemos en un entorno completamente *Google*, hemos aprovechado el analizador que viene con GAE. En la siguiente Figura podemos ver una captura de los resultados del análisis.

[2] Los resultados de esta tabla que no aparecen calculados, proceden de la terminal de GAE.

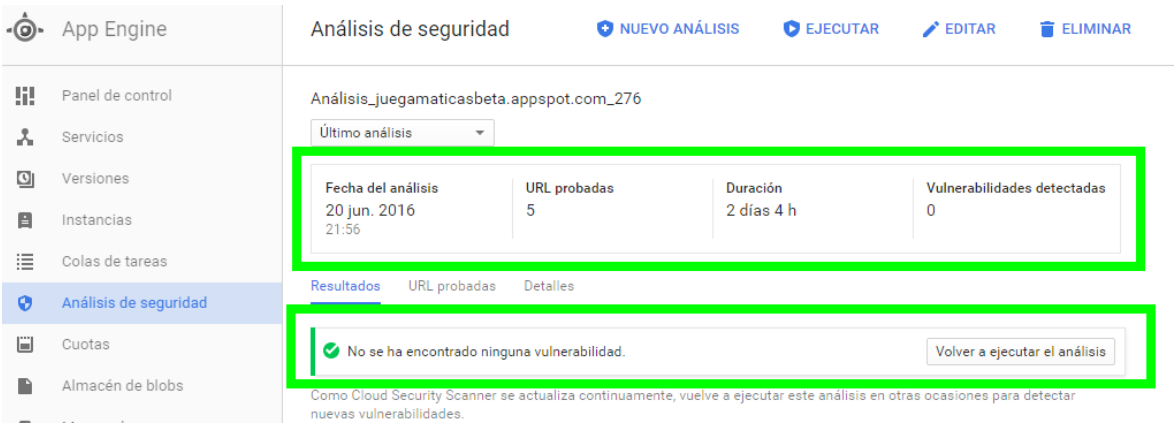

Figura 6-1 : Resultados del análisis de Seguridad GAE

6.1.5 Pruebas de aceptación

Juegamáticas ha seguido un desarrollo basado en prototipos (enseñábamos el prototipo, recibíamos las críticas y actuábamos en consecuencia), en el que cabría destacar dos tipos diferentes de pruebas de aceptación:

- **Con la educadora del *Colegio María Corredentora***: Ella ha sido quien nos ha guiado, y aceptando o rechazando los cambios que hemos ido haciendo sobre la aplicación. En la última reunión únicamente salieron mejoras para la siguiente versión de la aplicación. Aparte, se creó un test *Likert* para que el colegio valorase la aplicación, pero aún no hemos recibido respuesta.

- **Con usuarios *Síndrome de Down***: Dado que ellos son el usuario final al que está dirigida la parte más pesada de la aplicación (los ejercicios) nos parecía importante hacer alguna prueba con ellos, y que nos diesen algún tipo de *feedback*. En nuestro caso, participaron en las pruebas 2 estudiantes (una chica de 18 años que podemos ver en la Figura 6-2, y un chico de 13), que utilizaron la aplicación durante 10 minutos. De las pruebas sacamos únicamente sacamos una conclusión negativa (en el tipo de ejercicio 2 cambiamos algunas de las imágenes para que las figuras estuviesen más separadas entre sí), el resto de conclusiones fueron positivas. A la chica no hubo que guiarla ni explicarle nada, desde el primer momento entendió lo que había que hacer. Al chico, hubo que introducirle con el primer ejercicio, y a partir de ahí supo continuar él solo sin problemas.

Figura 6-2: M. utilizando *Juegamaticas*

Somos consciente que no es una muestra lo suficientemente grande o heterogénea para sacar conclusiones finales. En septiembre de 2016 tenemos planeados reunirnos otra vez en el *Colegio María Corredentora,* donde ya nos confirmado que podremos hacer nuevas pruebas con grupos más grandes y heterogéneos.

6.2 Resultados

Después de realizar estas pruebas, hemos sacado las siguientes conclusiones:

- Dado que vamos a continuar añadiendo nuevas funcionalidades, deberíamos empezar a desarrollar el módulo de test unitarios. Con ellos de base, iremos desarrollando test automáticos de integración utilizando Python, Selenium y *Behave* como framework de *testing* BDD (*Behaviour Driven Development*). Lo bueno de *Behave* es que los casos de uso (y casi los requisitos) se pueden transformar en tests utilizando lenguaje natural.

- Podemos escalar el número de usuarios al triple del máximo que había como requisito, sin tener que gastar dinero.

- Mientras no se encuentre y explote alguna vulnerabilidad en los servicios o infraestructura de *Google*, nuestra aplicación no debería estar en peligro. Aparte, continuaremos haciendo análisis de seguridad cuando vayamos a aplicar algún cambio de código.

- Debemos hacer más pruebas con grupos de usuarios finales, y hasta que ellos no den el visto bueno, no considerar que la aplicación está completamente finalizada.

7 Conclusiones y trabajo futuro

7.1 Conclusiones

El objetivo del proyecto era crear una aplicación de apoyo al aprendizaje matemático, dirigido a estudiantes con *Síndrome de Down*, prestando especial atención a la usabilidad, que fuese atractivo para ellos, y a que los ejercicios siguiesen una evolución acorde a las metodologías empleadas en el *Colegio María Corredentora*.

Todos estos objetivos han sido cumplidos satisfactoriamente, pero eso no significa que se haya terminado. Tenemos el compromiso, la convicción y las ganas de continuar con el trabajo durante el curso que viene, hasta que creemos la aplicación perfecta para los estudiantes y tutores (padres y profesores).

El único aspecto negativo que he visto durante la realización de este proyecto es que arece que *Google* ha dejado un poco descuidados los proyectos de GWT y Objectify en algunos aspectos (lo que hacía que a veces encontrar documentación fuese casi imposible, y por consiguiente hubiese que emplear mucho más esfuerzo), y sin embargo GAE ha mejorado considerablemente durante estos meses, enriqueciéndose en tecnologías a pasos agigantados.

7.2 Trabajo futuro

Dado que el trabajo tiene continuidad durante al menos el curso que viene, tenemos planeado:

- La activación de usuarios con rol Tutor debe hacerse de forma que no sea necesaria la intervención del Administrador.
- Parte de las funcionalidades actuales que sólo puede hacer el usuario Administrador, deben pasar al usuario Tutor.
- Las descripciones de los tipos de ejercicios (tanto escritas como sonoras) deben mejorarse.
- Ampliar la cantidad de tipos de ejercicio. Para la versión de Septiembre está pensado incluir 3 nuevos:
    - Tipo 7: Restas simples
    - Tipo 8: Suma de 2 cifras sin llevada.
    - Tipo 9: Resta de 2 cifras sin llevada.
- Aún no contamos con una descripción de qué datos de los alumnos deben poder verse o editarse en la vista "*Mis alumnos*" del usuario Tutor.
- Seguramente haya que ajustar los porcentajes con los que un usuario baja y sube de nivel.
- Realizar pruebas de usabilidad y aceptación con grupos de usuarios más grandes y genéricos.
- Realizar pruebas automatizadas que verifiquen las diferentes *releases* que vayamos haciendo.

Referencias

[2] Fuente: http://mariacorredentora.org/wpmcorg/
Visitado: 27/06/2016

[3] Fuente: https://play.google.com/store/apps/details?id=com.myfirstapp.series1.g
Visitado: 22/06/2016

[4] Fuente: https://itunes.apple.com/es/app/picaa/id373334470?mt=8
Visitado: 22/06/2016

[5] Fuente: http://eurekaapp.es/proyecto/
Visitado: 22/06/2016

[6] Fuente: https://fundaciongarrigou.org/our-causes/presentacion-aplicacion-eureka-y-eurolibro-en-cnmc/
Visitado: 22/06/2016

[1] Fuente: http://www.abc.es/20120905/familia-educacion/abci-aprendizaje-etapas-educacion-201209031121.html
Visitada: 14/04/2016

[7] Fuente: https://es.wikipedia.org/wiki/Cliente-servidor
Visitada: 15/06/2016

[8] Ideas extraídas de: http://lia.unet.edu.ve/avaunet/Glosario.htm
Visitada: 15/06/2016

[9] Fuente: https://ingsoftwarei2014.files.wordpress.com/2014/06/9j5wl.png
Visitada: 20/06/2016

[10] Fuente: https://www.adictosaltrabajo.com/tutoriales/cloudcomputing/
Visitada: 21/04/2016

[11] Fuente: https://cloud.google.com/appengine/docs/quotas#Mail
Visitada: 20/03/2016

[12] Fuente: http://www.gwtproject.org/doc/latest/tutorial/RPC.html
Visitada: 15/11/2015

[13] Fuente: http://ttsreader.com/es/
Visitada: 25/06/2016

[14] Fuente *NMAP*: https://nmap.org/
Visitada: 25/06/2016

[15] Fuente *Nessus*: http://www.tenable.com/products/nessus-vulnerability-scanner
Visitada: 25/06/2016

[16] *bugScout*: https://buguroo.com
Veracode: http://www.veracode.com/
Fortify: http://www8.hp.com/es/es/software-solutions/application-security/
Checkmarx: https://www.checkmarx.com/
Visitadas: 25/06/2016

Glosario

NoSQL	Not Only SQL
GAE	Google App Engine
GWT	Google Web Toolkit
POJO	Plain Old Java Object
BBDD	Bases de Datos
RPC	Remote Procedure Calls
CSS	Cascading Style Sheets
BDD	Behaviour Driven Development

Anexos

Anexo A: Resumen de la Metodología y principios de la enseñanza matemática del Colegio María Corredentora.

El colegio María Corredentora trabaja en la elaboración de una Metodología didáctica para la enseñanza de la Matemática desde hace algunos años. El equipo docente formuló como objetivo general: "Obtener mejores resultados en la competencia matemática de los alumnos del Centro". Tal desempeño iría conectado con los estándares que siempre han definido nuestro modelo educativo, así, se valoró necesario, la implementación de un nuevo modelo que apostara por la innovación educativa, lo que suponía poner esfuerzos en transformar el modelo educativo existente hasta ese momento hacia la construcción de un nuevo Paradigma educativo que reformulara el carácter que debía tomar la enseñanza de este área curricular.

El nuevo **Paradigma** se iba concretando con un talante innovador o de cambio, que tiene en cuenta, los siguientes aspectos:

- Conseguir "Claridad de conceptos, razonamiento correcto y capacidad para establecer relaciones".
- El docente debe dominar su materia y practicar la escucha activa, dirigiendo todos sus esfuerzos a que el alumno: sepa bien, quiera saber, se sienta bien sabiendo y aplique correctamente lo que sabe.
- Tener en cuenta que si el abuso de contenido incomprensible perjudica la acción formativa del individuo, la disminución de contenido que pueda comprenderse perjudica al desarrollo. Tanto error se comete cuando se intenta que un niño aprenda algo que supera su comprensión, como cuando se intenta disminuir la cantidad de conocimiento que pueda conseguir.
- Que las respuestas que obtenemos de los niños no coincidan con las que esperamos implica, simplemente, discrepancia entre la enseñanza y el aprendizaje, y no asegura que el niño tenga dificultad alguna para el aprendizaje de la Matemática. Más allá del término está su significado y, por tanto, el prejuicio de su diagnóstico: cuándo podemos hablar, o no, de dificultades en el aprendizaje de la Matemática; saber si esta dificultad es primaria o duradera, o bien secundaria. Son muchos los investigadores y estudiosos del tema los que agregan un problema importante y frecuente en su diagnóstico: la enseñanza inadecuada.
- Que el profesor enseñe y los alumnos aprendan lo que el profesor enseña, sólo tiene aprobación y vigencia cuando lo aprendido desarrolla el pensamiento matemático. La pregunta fundamental no es ¿qué hay que enseñar?, sino ¿qué conseguimos con lo que estamos enseñando?
- La fiabilidad de lo que un profesor enseña, se mide por la validez de lo que sus alumnos son capaces de hacer sin él.
- No todos los niños tienen la misma capacidad para aprender matemáticas, pero sí todos tienen la misma necesidad de aprenderlas. La tarea escolar consiste en cubrir las necesidades, y no en clasificar capacidades.
- Los materiales que podremos utilizar para la enseñanza de la Matemática son muchos, pero no apoyarán éstos su eficacia en las propiedades que poseen, sino en su posibilidad para interactuar en la mente del sujeto y que éste pueda: formular, y

suponer, y descubrir, y comprender e interpretar correctamente. Sin olvidar como otros materiales: la realidad y la evidencia.

Materiales

No se puede eludir a estas edades el trabajo con materiales y recursos concretos, pues es a través de lo experimental desde donde se genera el proceso que nos permite pasar de lo particular, a la generalización intelectual de una dinámica de relaciones. La utilización correcta de material manipulativo enriquece los procesos de enseñanza-aprendizaje, aumentando el rendimiento en la comprensión y aplicación de los conceptos matemáticos. El rol del profesor es de suma importancia en la toma de decisiones para la utilización de material. Según objetivos y funciones, tendrá en cuenta aspectos didácticos, aplicativos, técnicos,... y de organización de espacios, tiempos y agrupamientos. No se debe olvidar que el uso de materiales y recursos es un medio para la adquisición del conocimiento matemático; de nada servirían al aprendizaje, si ese conocimiento no goza de esclarecida evidencia para la enseñanza.

Una vez conocido qué hay que aprender, buscaremos los medios que lo faciliten, dando en todo momento al aprendizaje un significado de aplicación práctica eficaz; no se trata tanto de rellenar páginas, manipular, manejar programas o navegar, como de utilizar el tiempo para vivir realidades educativas de ámbito científico y cultural, que supongan verdadera innovación en la enseñanza de las Matemáticas y en su extensión a otras áreas de conocimiento.

El Centro continúa elaborando actividades coherentes a estos principios, recogidos en materiales que manifiestan físicamente, los logros que los alumnos van conquistando en el aprendizaje de la matemática. El uso de estos materiales es fundamental para el desarrollo del pensamiento matemático, y su uso, debe ser estructuralmente secuenciado y adaptado a las características personales de los alumnos. Se dividen en: **Materiales manipulativos e impresos**.

Contenidos

Siguiendo una línea de contenidos secuenciados, el alumno experimentará a lo largo de la etapa escolar todas las experiencias necesarias que permitan poner de manifiesto la consecución de los objetivos para cada uno de los contenidos que vaya trabajando. Ahora bien, la graduación por objetivos de cada uno de los objetivos de un mismo contenido, no implica la limitación de experiencias para otros contenidos. De manera que un alumno que esté trabajando, poniendo el ejemplo, clasificaciones, no se le debe privar de paralelamente tener experiencias de cualquier atributo de los objetos, poniendo el ejemplo, tamaño. Si bien es cierto que hay contenidos que para su abordaje, necesitan unos prerrequisitos (para multiplicar es necesario que el alumno sepa sumar), en muchas otras ocasiones, no se hace necesario esta relación dependiente y jerarquizada. La adquisición de conocimientos posee un estado de grados de comprensión y las actividades propuestas tendrán que contemplarlos. El proceso, por ejemplo: “estimación, aproximación, precisión” si se corresponde a un estado ligado y no un conjunto de ejercicios aislados y desprovistos de significado.

La temporalización que se hace sobre los contenidos matemáticos no debe depender tanto de los propios contenidos, como sí del nivel de desarrollo que tenga el alumno. En muchos

casos el alumno puede estar trabajando numeración bajo una estructura de composición y descomposición, poniendo el ejemplo hasta el número 6, y paralelamente ir perfeccionando la técnica de conteo hasta el número 30. La actividad mental que el alumno hace en el primer caso es muy distinta a la del segundo desempeño. En matemáticas no hay temas, sino estructuras. La mente humana, por la estructura cerebral que tenemos, es capaz de afrontar situaciones de diferente complejidad por el mero hecho que ciertas habilidades matemáticas dependen de señaladas zonas de aprendizaje que el alumno puede llegar a desarrollar de manera exitosa. El cerebro expresa un dominio de desarrollo de cero a seis años que no se repetirá con el mismo esplendor a lo largo de nuestra vida. Si a esto añadimos el deseo hiperactivo por descubrir y el enorme potencial de vida activa y afectiva que se puede desplegar, la capacidad de aprendizaje a esas edades es incalculable.

Esa capacidad de aprendizaje debe estar íntimamente unida a una gran capacidad de enseñanza. Incorporar a la mente del niño un conjunto de términos y representaciones incomprensibles perjudica su acción formativa, pero la disminución de contenido que pueda comprenderse perjudica al desarrollo; tanto error se comete cuando intentamos que un niño aprenda algo que supera su comprensión, como cuando disminuimos la cantidad de conocimiento y facilitamos el esfuerzo intelectual al que un niño hubiera podido llegar. Por eso, actualizarse no consiste en imitar procedimientos que están de moda, sino en conseguir, en tiempo real y con los niños actuales, los objetivos dirigidos a la adquisición del conocimiento y el desarrollo personal.

Metodología didáctica

La intervención didáctica aportará un significado de utilidad en la adquisición del conocimiento matemático, en la medida que favorezca la práctica de la investigación y el descubrimiento, para la consolidación y aplicación de los conceptos.

El uso de la pregunta como soporte didáctico, para presentarle al niño desafíos que estimulen la investigación y aseguren el descubrimiento de los conceptos y relaciones, es esencial en cualquier proceso de enseñanza que genere un aprendizaje válido de las Matemáticas. Del mismo modo, es importante invitar al alumno a hacerse preguntas, como medio fundamental para la adquisición de conocimientos.

La necesidad educativa actual supone tener en cuenta: los elementos de diversidad del alumnado; la atención al vocabulario del alumno como punto de partida; la configuración cíclica de los contenidos; una secuenciación precisa, sabiendo qué va antes y qué va después, como conceptos previos necesarios para la comprensión de otros posteriores; la aportación de mecanismos de control que favorezcan la autocorrección; la necesidad de incorporar el razonamiento lógico al estudio de las Matemáticas; y, la planificación cuidadosa de pequeños retos y desafíos que estimulen el interés y el esfuerzo por el aprendizaje. Se atenderá en primer lugar, desde el respeto al conocimiento científico, a la elaboración intelectual de las ideas, para terminar con la utilización precisa del símbolo y la terminología matemática.

Anexo B: Modelado

En este anexo explicaremos un poco más a bajo nivel cómo funciona la aplicación.

Sistema

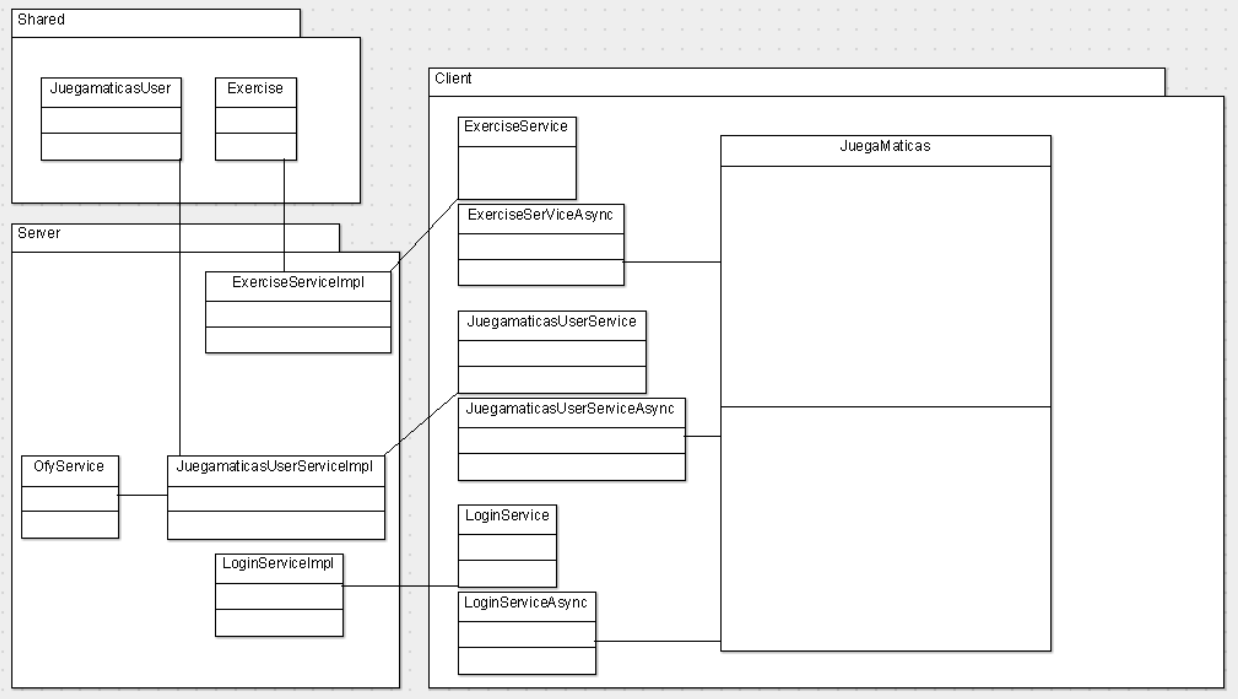

Figura AnB-1: Representación del sistema

- En parte *Shared*, tenemos los objeto que queremos comunicar del servidor al cliente. Están en este módulo ya que ambas partes tienen que saber qué tiene el modelo.

- Como hemos implementado la comunicación mediante RPC, tenemos que crear 3 elementos:
  - <*nombreServicio*>*Service.java*: Interfaz que representa al servicio (con los métodos que vayamos a utilizar). En nuestro caso las tenemos en la parte cliente, pero esto no es estrictamente (*JuegamaticasUserService.java*, *ExerciseService.java*, *LoginService.java*).
  - <*nombreServicio*>*ServiceImpl.java*: Implementación del servicio en la parte servidor.En nuestro caso tenemos *JuegamaticasUserServiceImpl.java y ExerciseServiceImpl.java*. No ha sido necesario implementar *LoginServiceImpl.java* ya que lo que hacemos es llamar a ese servicio de Google.
  - <*nombreServicio*>*ServiceAsync.java*: Interfaz asíncrona que será llamada desde la parte cliente. En nuestro caso tenemos *JuegamaticasUserServiceAsync.java, ExerciseServiceAsync.java, LoginServiceAsync.java*

- *JuegamaticasUserServiceImpl.java* es donde implementaremos las llamadas al servicio *Objectify* (*Ofy.java*).

Objetos

Los objetos principales que hemos modelado son:

- **JuegamaticasUser**: La representación del usuario de la aplicación. Es el único elemento que guardamos en base de datos, ya que dentro del usuario tendremos diferentes conjuntos para guardar qué ejercicios ha hecho y con qué resultados. Cuando hacemos login por primera vez, recogemos el correo del nuevo usuario y se lo adjudicamos como *userId*. En un futuro seguramente necesitemos que haya diferentes niveles de tutores, por lo que *is_tutor* lo cambiaremos por un enum.

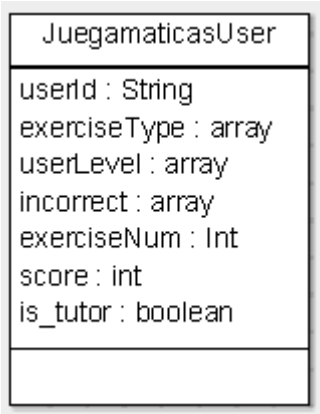

Figura AnB-2: Modelo User

- **Exercise**: Es la representación de los ejercicios que va a solucionar un usuario alumno. Utilizamos campos String para que luego no haya que transformarlos en la parte cliente, aunque en la parte servidor sí que haya que transformarlos.

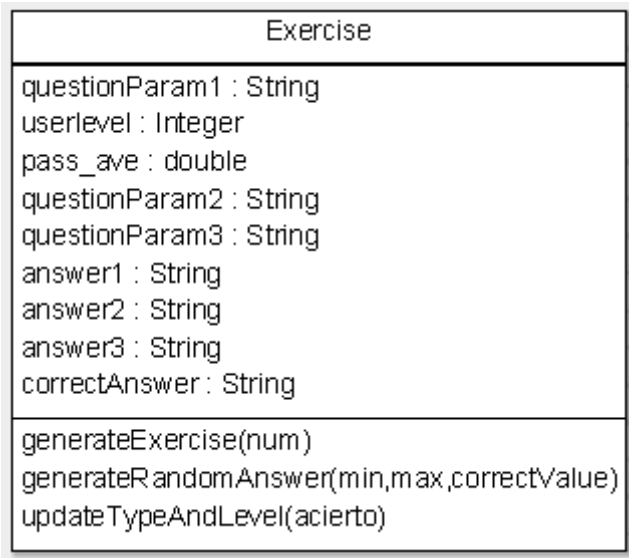

Figura AnB-3: Modelo Exercise

• **Juegamaticas**: Es la clase principal del cliente. Tiene los atributos base para pintar los ejercicios y los métodos para hacerlo. Además, es la encargada de ir llamando a los diferentes servicios y de reproducir los sonidos.

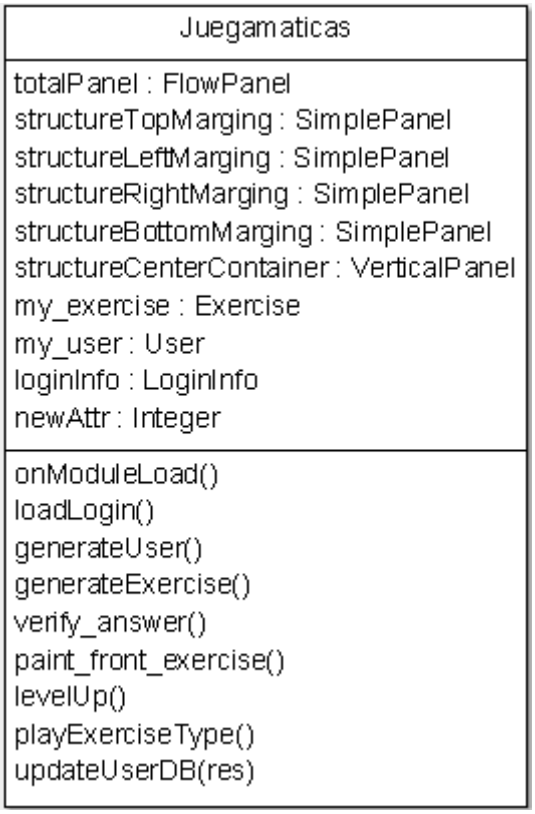

Figura AnB-4: Modelo Juegamaticas

Anexo C: Tipos de Ejercicios

En este anexo encontraremos capturas de los diferentes tipos de ejercicio que existen ahora mismo en la aplicación. Estas capturas son de una versión que no está destinada a los usuarios finales, los estudiantes, de ahí que aparezca información como: tipo de ejercicio, nivel, puntuación, correo del usuario (información muy útil para que el personal no técnico que tiene que validarla pueda entender cómo funciona).

- **Tipo 0**: ***Asociación del mismo número de figuras iguales*.**

Figura AnC-1: Ejercicio tipo 0

- **Tipo 1**: ***Asociación del mismo número de figuras distintas*.**

Figura AnC-2: Ejercicio tipo 1

- **Tipo 2**: ***Asociación de guarismos a número de figuras*.**

Figura AnC-3: Ejercicio tipo 2

- **Tipo 3**: ***Asociación de número de figuras a guarismos*.**

Figura AnC-4: Ejercicio tipo 3

- **Tipo 4**: ***Parejas fundamentales (figuras + guarismos)*.**

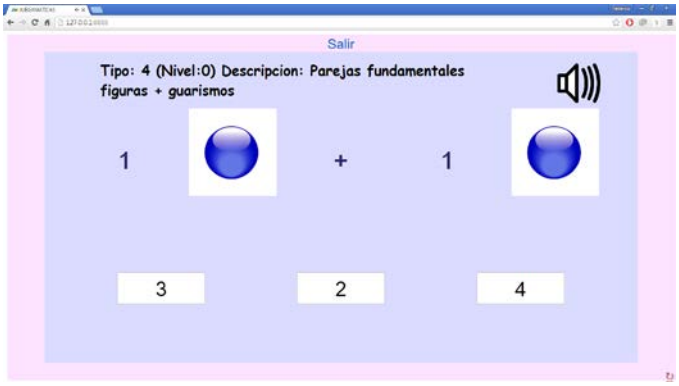

Figura AnC-5: Ejercicio tipo 4

- **Tipo 5:** ***Parejas fundamentales (guarismos)*****.**

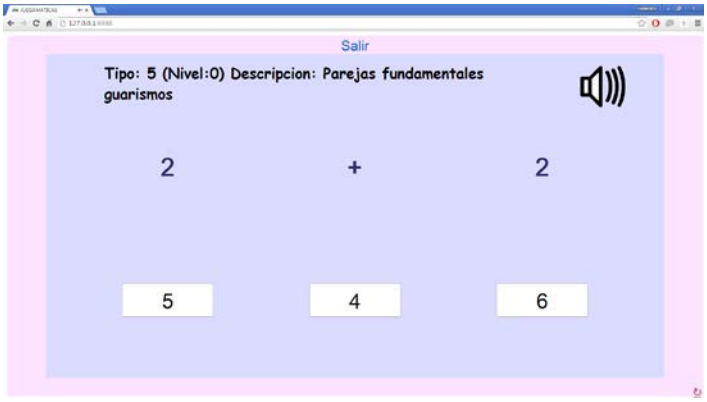

Figura AnC-6: Ejercicio tipo 5

- **Tipo 6**: ***Operaciones con guarismos*****.**

Figura AnC-7: Ejercicio tipo 6

Anexo D: Algoritmo de evaluación

Recordemos que para decidir si un alumno debía subir o bajar de nivel, seguíamos las siguientes reglas:

- Si el Alumno tiene un porcentaje de acierto mayor al 80% en un nivel, pasa al siguiente. Si está en el nivel máximo dentro de un tipo de ejercicio, pasará al siguiente tipo de ejercicio.
- Si el Alumno tiene un porcentaje de acierto menor al 50.1%, bajará de nivel. Si está en el nivel mínimo dentro de un tipo de ejercicio, bajará de tipo de ejercicio.

En la clase Exercise existe el método *updateTypeAndLevel*, que es el que decide si se pasa de nivel o de tipo de ejercicio:

```
private void update_type_and_level(int acierto) {
        //Calculamos dónde están los resultados de nivel del usuario
        int level_info_index = 2 * userLevel;
        int num_ok = this.level_info.get(level_info_index); //Numero de aciertos en ese nivel
        int num_total = this.level_info.get(level_info_index+1);  // Numero de ejercicios hechos en ese nivel

        if(acierto==1){
        //Si entra un acierto, incrementamos el numero de ejercicios hechos correctamente, y el numero de ejericios totals
                this.level_info.set(level_info_index, num_ok+1);
                this.level_info.set(level_info_index+1, num_total+1);
        }else if (acierto==0){
                //Si entra un error, incrementamos únicamente el numero de ejercicios total
                this.level_info.set(level_info_index+1, num_total+1);
        }

        //Calculamos el porcentaje de acierto en el nivel actual.
        double pass_average =
((double)(this.level_info.get(level_info_index)))/((double)(this.level_info.get(level_info_index+1)));
        pass_average = pass_average*100.0;

        if (pass_average > 80.0){
                if(this.userLevel < 5){
                //Si el usuario tiene un porcentaje mayor al 80%, y no está en el último nivel, subirá de nivel
                        this.userLevel++;
                }else if(this.exerciseType < 6){
                //Si el usuario puede pasar the tipo de ejercicio, pasará de tipo de ejercicio
                        this.exerciseType++;
                        for(int aux=0; aux<12; aux++){
                                this.level_info.add(aux, 0);
                        }
                        this.userLevel = 0;
                }
        }else if(pass_average < 50.1){
                if(this.userLevel > 0){
                //Si el usuario tiene un porcentaje menor al 50.1%, y no está en el primer nivel, bajará de nivel
                        this.userLevel--;
                }else if(this.exerciseType > 0){
                //Si el usuario puede bajar de tipo de ejercicio, bajará
                        this.exerciseType--;
                        for(int aux=0; aux<12; aux++){
                                this.level_info.add(aux, 0);
                        }
                        this.userLevel = 0;
                }
        }
}
```

Anexo E: Casos de uso

En este anexo mostraremos algunos de los casos de usos creados para este proyecto:

CU-01	Registrar un Alumno
Dependencias	RNF-1
Precondición	· El usuario debe tener una cuenta Google. ·El usuario debe ser nuevo en el sistema o haber sido eliminado con anterioridad
Descripción	Al acceder un nuevo usuario en el sistema, debe registrarse en la BBDD
Secuencia	El nuevo usuario accede a la URL El usuario hace click en el botón "Entrar" de la página de bienvenida. El usuario Introduce sus credenciales en la página de Google.
Postcondición	Aparece una nueva entrada en la tabla de usuarios de Objectify

CU-02	Alumno supera correctamente un ejercicio
Dependencias	RF-3, RF-4, RF-7
Precondición	· El usuario debe estar logueado en la aplicación
Descripción	Al resolver correctamente un ejercicio se escuchará una notificación sonora agradable, y se cambiará el ejercicio.
Secuencia	El usuario contesta correctamente un ejercicio
Postcondición	Sonará *yes* Aparecerá automáticamente un nuevo ejercicio.

CU-03	Alumno falla un ejercicio
Dependencias	RF-3, RF-4, RF-5, RF-7
Precondición	· El usuario debe estar logueado en la aplicación
Descripción	Al resolver equivocadamente un ejercicio, se escuchará una notificación menos agradable, y el botón pulsado cambiará de color.
Secuencia	El usuario contesta erróneamente a un ejercicio
Postcondición	Sonará *No* El botón elegido erróneamente: a. Si la respuesta era con guarismos: Se pondrá rojo entero b. Si era una figura: Aparecerá un grueso marco rojo en el botón.

CU-04	Alumno supera un tipo de Ejercicio
Dependencias	RF-7, RF-8, RF-12, RNF-7
Precondición	·Usuario debe estar logeado y en el primer ejercicio de un tipo de ejercicio
Descripción	Al superar un tipo de ejercicio, cambiará la interfaz del ejercicio ajustándose al nuevo tipo, y escucharemos una notificación sonora.
Secuencia	El usuario superará correctamente los ejercicios que le vayan surgiendo. Dependiendo de sus estadísticas necesitará resolver más o menos ejercicios.
Postcondición	Superados el número necesario de ejercicios, sonará una atractiva notificacón y la interfaz del ejercicio será sustituida por la interfaz del nuevo tipo.
Comentario	Si utilizamos un usuario que no tenga estadísticas en un tipo de ejercicio, podremos superar el tipo de ejercicio acertando 6 veces seguidas.

CU-05	Alumno desciende de tipo de ejercicio
Dependencias	RF-7, RF-8, RF-12, RNF-7
Precondición	·Usuario debe estar logeado y en un ejercicio que no pertezca al set de ejercicios tipo 0.
Descripción	Al descender en un tipo de ejercicio, cambiará la interfaz del ejercicio ajustándose al nuevo tipo.
Secuencia	El usuario superará correctamente los ejercicios que le vayan surgiendo, para ello siempre que le venga un ejercicio, primero fallará y luego corregirá. Dependiendo de sus estadísticas necesitará resolver más o menos ejercicios.
Postcondición	Superados el número pertinente de ejercicios la interfaz del ejercició será sustituida por la interfaz del nuevo tipo.
Comentario	Si utilizamos un usuario que no tenga estadísticas en un tipo de ejercicio, podremos superar el tipo de ejercicio acertando 6 veces seguidas.

CU-06	Alumno sale de la aplicación, y vuelve a acceder
Dependencias	RNF-1, RF-1
Precondición	El usuario debe estar registrado y logeado.
Descripción	Al salir del sistema y volver a entrar en el sistema, el usuario seguirá estando en el mismo tipo de ejercicio y nivel.
Secuencia	El usuario conoce en qué tipo de ejercicio y nivel está El usuario hace click en el botón "Salir" de la vista de ejercicios El usuario vuelve a loguearse en la aplicación
Postcondición	El usuario comprobará que sigue en el mismo tipo de ejercicio y nivel.

Anexo F: Pruebas en dispositivos

Para hacer las pruebas con los diferentes dispositivos utilizamos una herramienta de desarrollo de *Google Chrome*, que permite emular diferentes dispositivos. El problema de esta herramienta es que si hacemos que se gire la pantalla, hace que no funcione bien el *responsive* de la aplicación (lo que hace en realidad es coger las medidas anteriores y aplicarlas regular, de manera que los elementos que utilizan *css* dinámico se restablecen mal). Elegimos dispositivos más o menos modernos, y que tuviesen diferentes resoluciones:

- **Original (*Chrome*)**

Figura AnF-1: Ejecución en Google Chrome

- **Nexus 4**

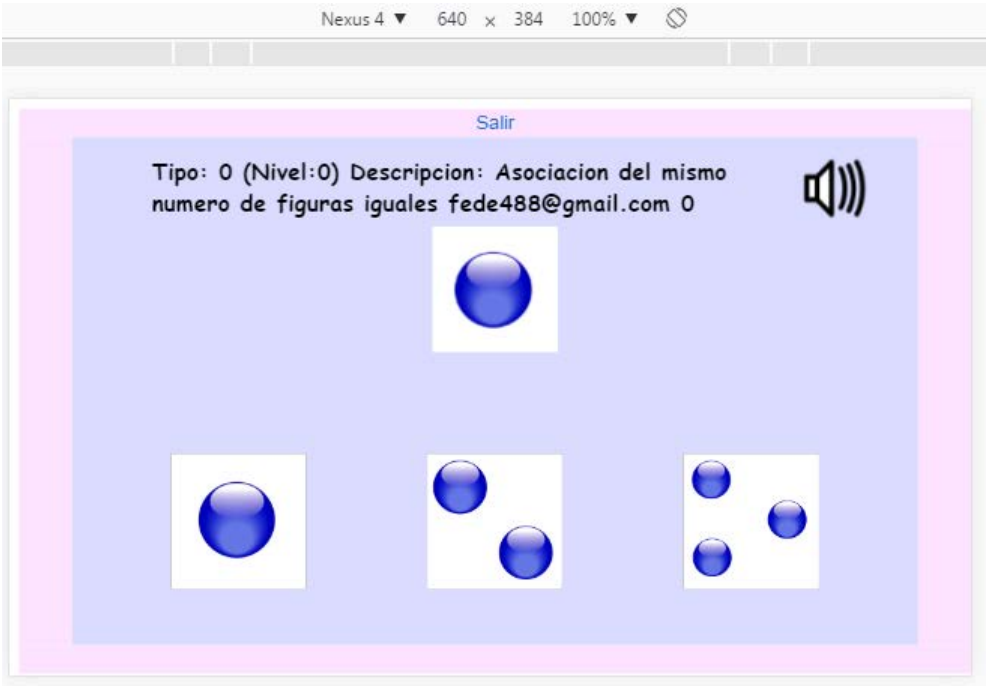

Figura AnF-2: Emulación en Nexus 4

- **Galaxy S5**

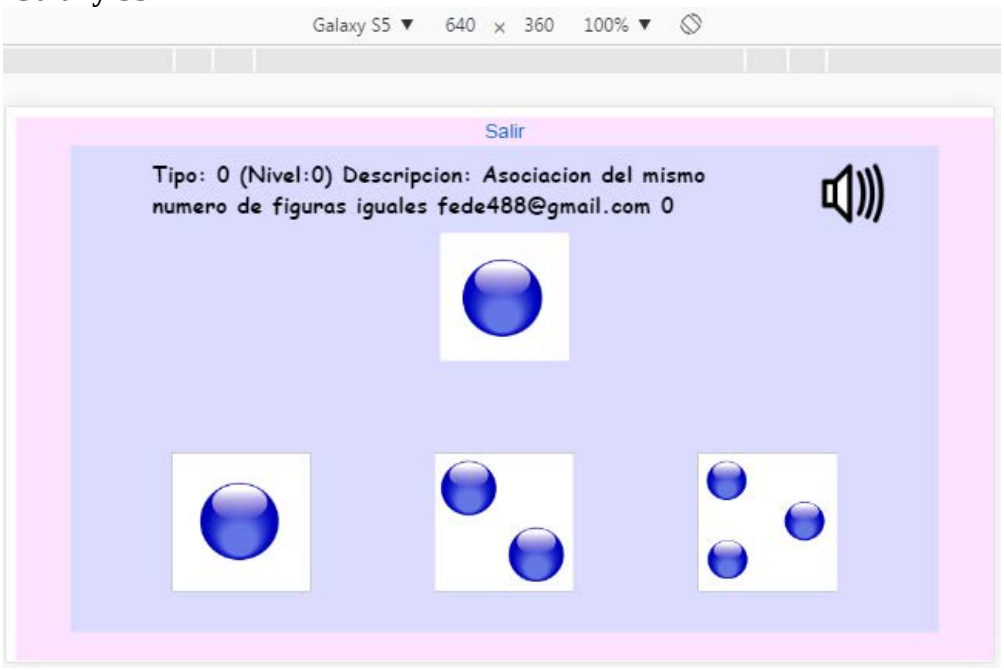

Figura AnF-3: Emulación Samsung Galaxy S5

- **Iphone 5**

Figura AnF-4: Emulación IPhone 5

- **Iphone 6 plus**

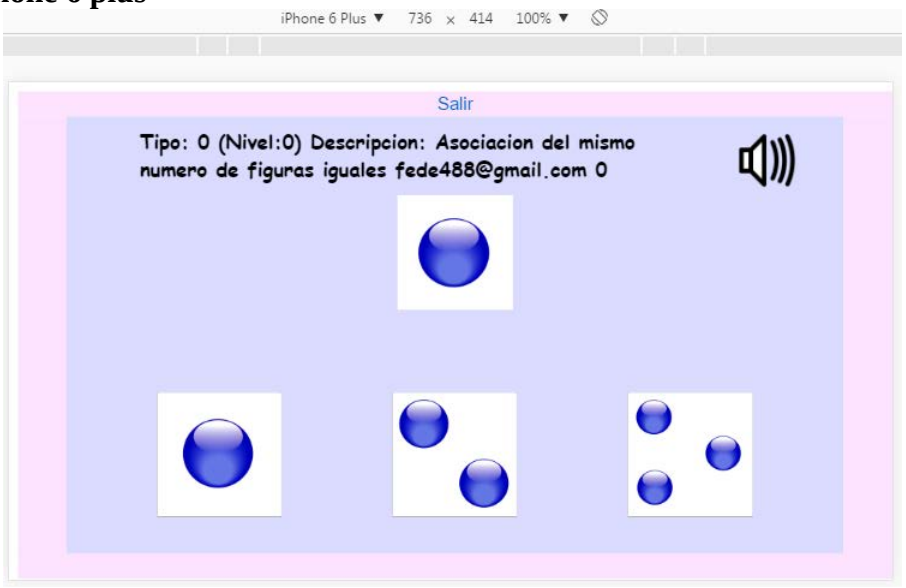

Figura AnF-5: Emulación IPhone 6 Plus

- **Ipad**

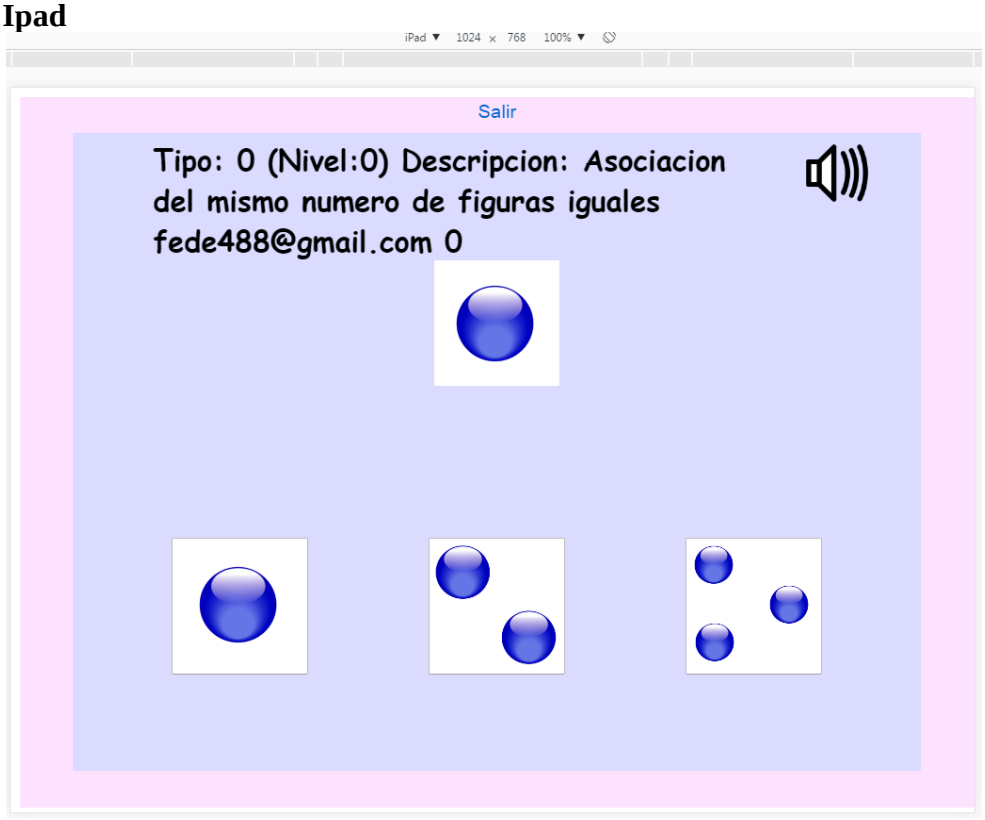

Figura AnF-6: Emulación IPad

Haciendo diferentes pruebas con dispositivos físicos, nos dimos cuenta de que si ponemos una descripción del ejercicio muy larga (como la que se ve en las anteriores capturas, que tiene información para *debug*), podía descomponerse. De ahí que uno de los puntos de trabajo futuro sea modificar esas descripciones para hacerlas lo más cortas y claras posibles (estamos a la espera de que el colegio nos mande los nuevos textos).

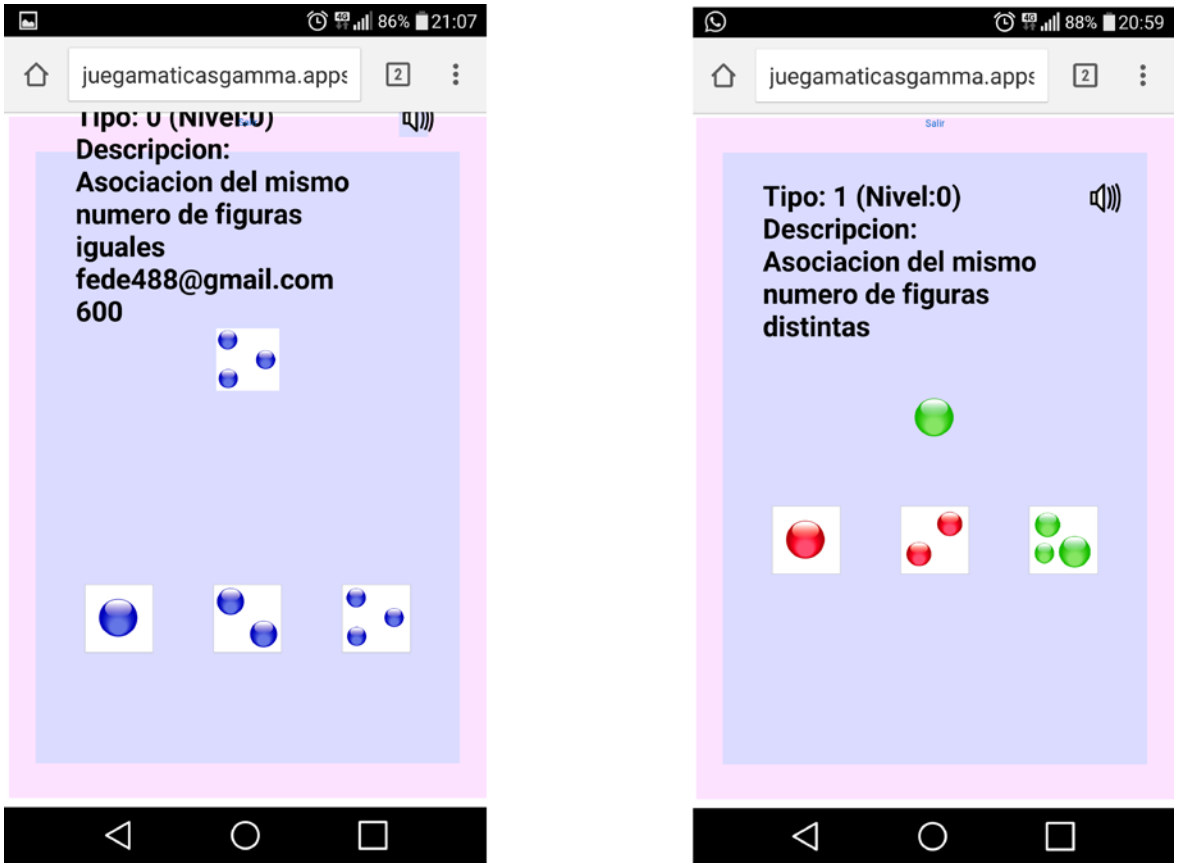

Figura AnF-7: Capturas LG G4

También descubrimos, que **IE** no es del todo compatible con algunas características de Java, lo que puede hacer que se pierdan posiciones de los elementos:

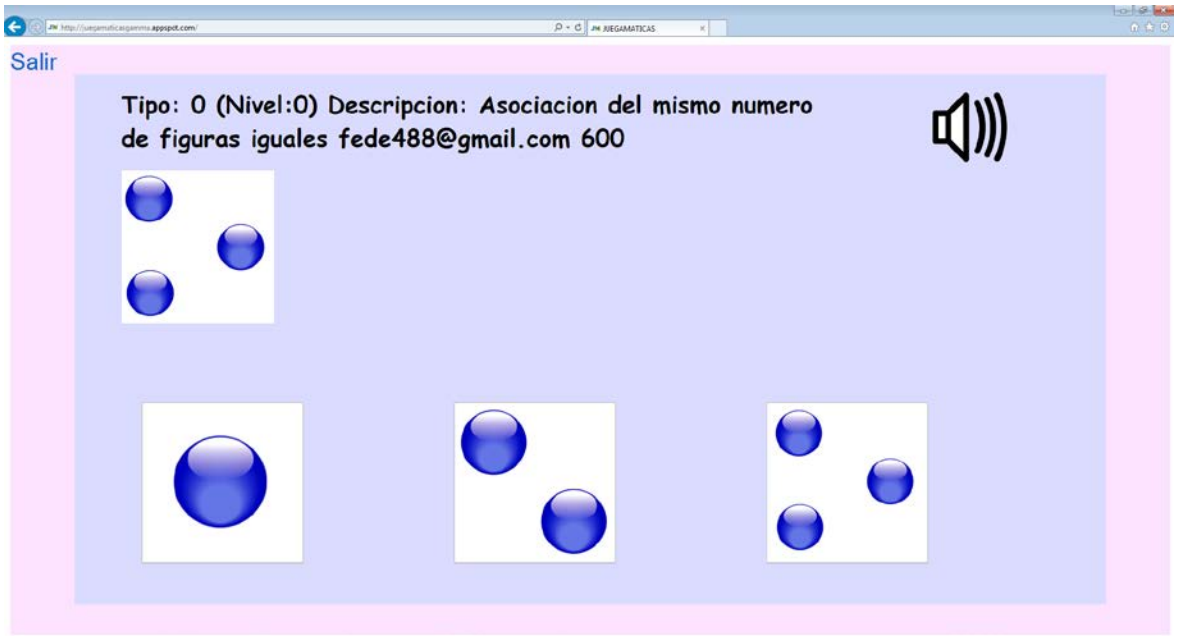

Figura AnF-8: Captura IE 11

Anexo G: Manual de la aplicación

Este manual de uso está dirigido a usuarios finales de la aplicación a día 26/06/2016. La aplicación puede sufrir modificaciones en tiempos futuros.

Administrador

Login: El usuario administrador accederá a la aplicación mediante la url http://juegamaticasbeta.appspot.com/ donde encontrará un mensaje de acceso y bienvenida que le llevará a una vista en la que podrá introducir sus credenciales (si es su primer acceso).

Cuenta Administrador: Después del login, el administrador deberá enviar un correo a soportejuegamaticas@gmail.com, desde la cuenta con la que ha entrado en la aplicación, en el que deberá incluir:

- Asunto: Administrador Juegamáticas
- Lista de correos de los usuarios a los que quiere tutelar.
- Explicar la relación que guarda con los alumnos a tutelar (parentesco, profesorado, etc.).

La solicitud se resolverá en un plazo máximo de una semana, y se notificará mediante un correo electrónico a esa cuenta. Si no queda clara la relación que guarda con los alumnos a tutelar, se le notificará que no ha sido posible y por qué.

Una vez esté activada la cuenta de Administrador, el usuario accederá directamente a la vista "*Mis alumnos*" desde la que podrá gestionar a los alumnos tutelados

Alumno

Login: Idéntico al caso Tutor. La idea es que el primer login se haga acompañado por su Tutor, y siempre y cuando no se cierre sesión, no tendrá que volver a hacer login en ese dispositivo.

Ejercicios: Las vistas de ejercicios, aunque cambian su contenido, tienen la misma funcionalidad en todos los casos:

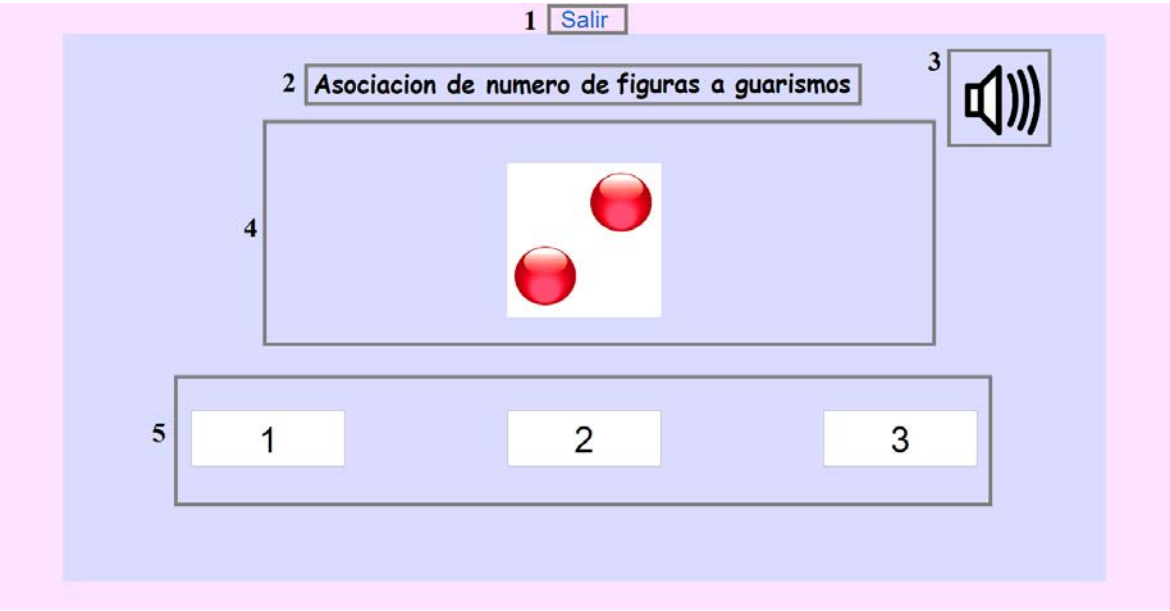

Figura AnG-3: Resumen de los elementos de un ejercicio

1. Botón de salida de la aplicación: Al presionar este botón, se cerrará la sesión de usuario. Si el usuario va a trabajar con la aplicación desde el mismo dispositivo, la idea es que no tenga que pulsarlo nunca, ahorrándose con ello tener que hacer login la próxima vez que acceda a la aplicación.
2. Descripción escrita del tipo de ejercicio.
3. Botón de reproducción de descripción del ejercicio: Al presionar este botón sonará la descripción del ejercicio con una voz clara y pausada.
4. Enunciado del problema: Dependiendo del tipo de ejercicio en el que se encuentre el usuario (en la figura A-3 se trata de un ejercicio tipo 3) los enunciados irán variando (pueden mostrar figuras o guarismos o mezclas de ambas). El enunciado muestra el ejercicio a mostrar en ese momento.
5. Resultados del problema: Al igual que los enunciados, los resultados variarán según el tipo de ejercicio en el que se encuentre el alumno en ese momento. Siempre aparecen 3 posibles soluciones a elegir una. En caso de que el usuario falle, se marcará la opción como incorrecta y podrá volver a intentar resolverlo.

Los usuarios, una vez dentro de la aplicación, no deben preocuparse por tiempos ni de cerrar sesión ni nada parecido. Pueden dejar de hacer la actividad en cualquier momento, y retomarla cuando quieran.